Assignment 2 - Understanding Promotions

Summary	To be able to analyze marketing data using Salesforce Einstein analytics studio
Author	Rishvita Bhumireddy, Pranathi Manusanipalli , Jugal Sheth
Category	Web
Codelab Link	https://codelabs-preview.appspot.com/?file_id=16_qjTsxV4dfyPo2Ni60gBfFIXtjn0Sk-IKDV99dqyTg#0
Youtube Link	https://youtu.be/iis1fu0tck0

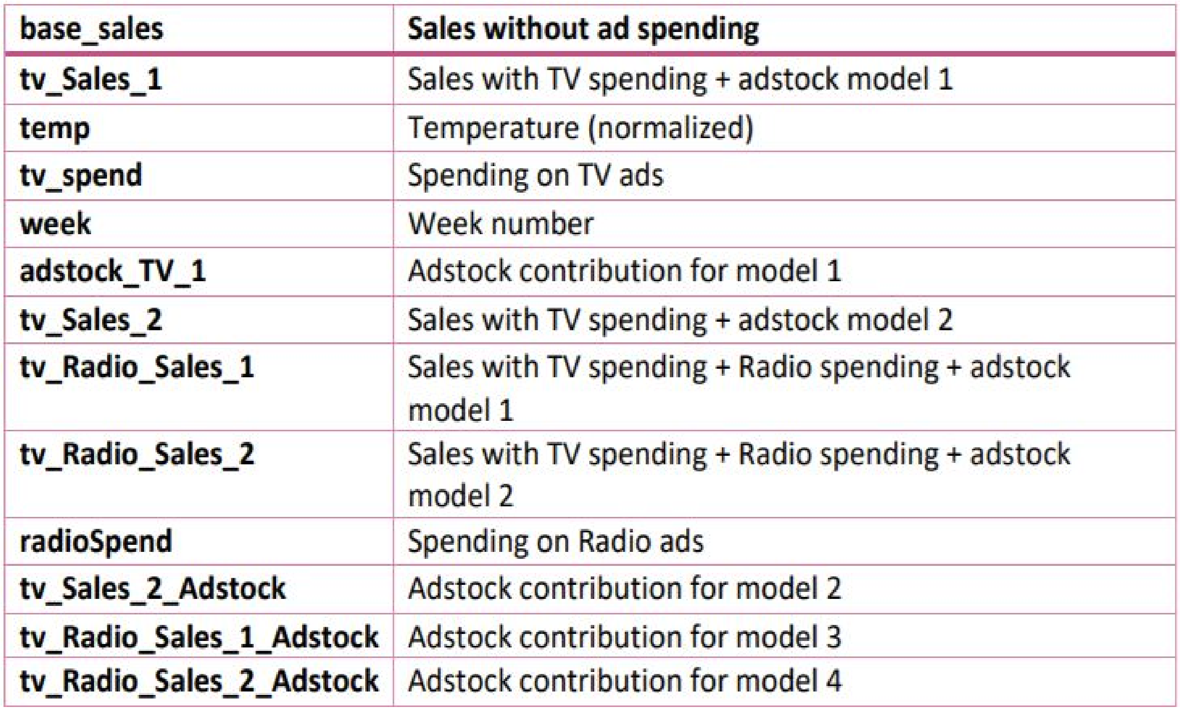
Introduction

Case:

Marketa analytics have weekly historical data and have built a model that shows sales as a function of temperature, tv ad spending, radio spending and adstock effects.

Data:

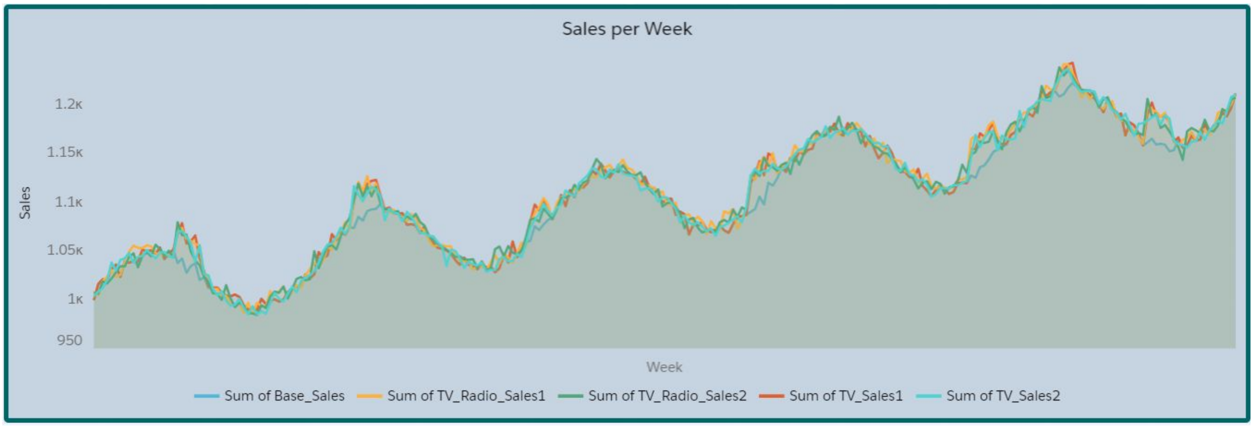
We are using a Dataset which comprises of Sales and spending by Tv and Radio mediums



base_sales	Sales without ad spending
tv_Sales_1	Sales with TV spending + adstock model 1
temp	Temperature (normalized)
tv_spend	Spending on TV ads
week	Week number
adstock_TV_1	Adstock contribution for model 1
tv_Sales_2	Sales with TV spending + adstock model 2
tv_Radio_Sales_1	Sales with TV spending + Radio spending + adstock model 1
tv_Radio_Sales_2	Sales with TV spending + Radio spending + adstock model 2
radioSpend	Spending on Radio ads
tv_Sales_2_Adstock	Adstock contribution for model 2
tv_Radio_Sales_1_Adstock	Adstock contribution for model 3
tv_Radio_Sales_2_Adstock	Adstock contribution for model 4

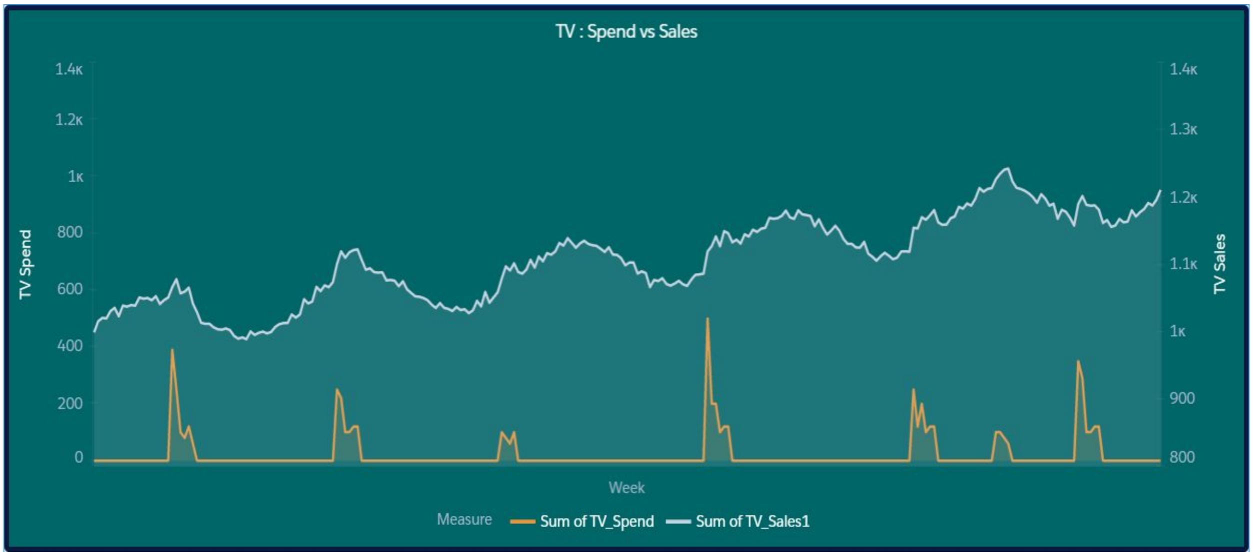
How is sales related to week numbers?

We observe a seasonal increasing trend, as the weeks increase there is an increase in sales overall, when looking at the granular level, the seasonal trends give us a cyclical range of 45-50 weeks.



How does TV spending affect sales? Can you quantify it?

As there is increase in spending the sales also increase and eventually because of adstock modelling there is an overall increase in sales, according to the trend observed



Quantifying the Increase in Sales due to TV Ads:

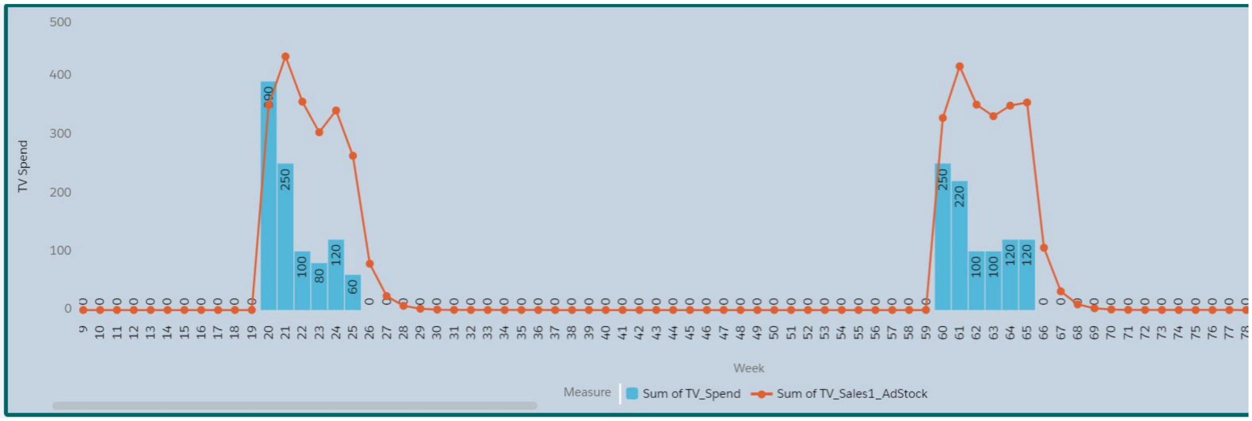
Sales occurred only due to TV Ads

tv_Sales_1- base_Sales
=287987.83-286827.62
=1160.21

On the whole the sales has increased 1160.21 units compared to base sales due to TV Ad Spending. The average increase in sales due to TV Ad spend is 4.46 units .

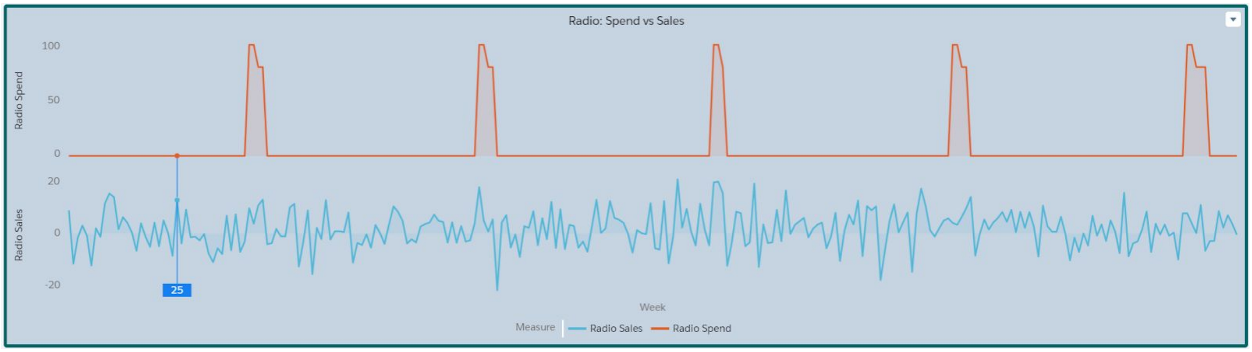
Are there any adstock effects for TV ad spending?

- Looking at the Chart, there is a slight increase in sales and this effect stays till a certain amount of time. From the graph below , as there are more spending cycles, the infinitesimal increase in sales is showcased as an increasing trend
- Thus there seems to be an adstock effect on TV as spending



How does Radio spending affect sales? Can you quantify it?

- Observing the chart tells us that there is a very random effect of spending on radio ads and since there is no fix pattern, we can safely assume there is no correlation between radio spend and sales trend.



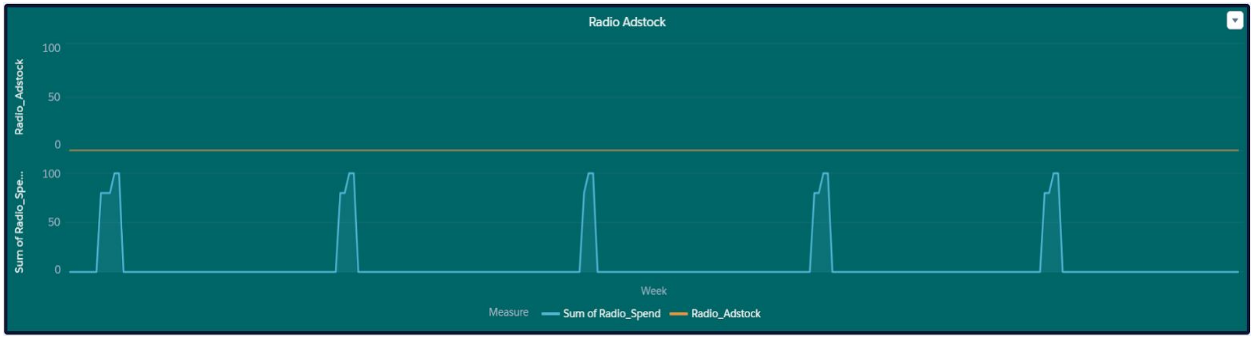
Quantifying the Increase in Sales due to Radio Ads:

Sales occurred only due to Radio Ads
TV_Radio_Sales1_Adstock - TV_Sales1
=288435.01 - 287987.83
=447.17

On the whole the sales has increased 447.17 units compared to base sales due to Radio Ad Spending. The average increase in sales due to TV Ad spend is 1.71 units

Are there any adstock effects for radio ad spending?

- The chart clearly shows that there is no adstock effect for Radio ads



Part 2:

Introduction:

Marketa Analytics came across this blog and is intrigued to analyze various Attribution models and determine the ideal model that provides better ROI for the advertiser and reduced ad exposure for the user.

Data Description:

This dataset represents a sample of 30 days of Criteo live traffic data. Each line corresponds to one impression (a banner) that was displayed to a user. For each banner we have detailed information about the context, if it was clicked, if it led to a conversion and if it led to a conversion that was attributed to Criteo or not. Data has been sub-sampled and anonymized so as not to disclose proprietary elements.

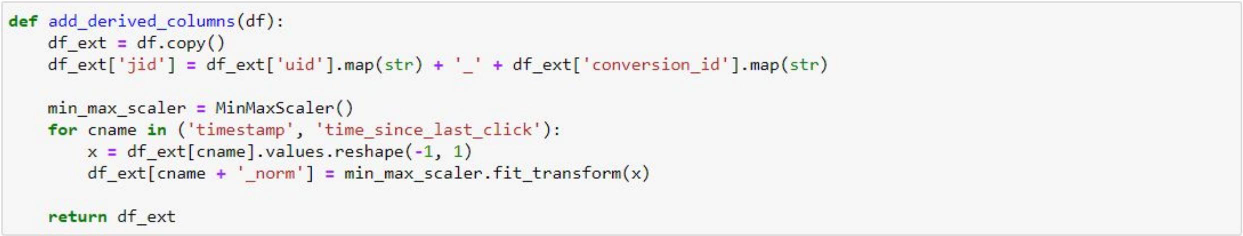
Here is a detailed description of the fields

- Timestamp: timestamp of the impression
- UID: unique user identifier
- Campaign: unique campaign identifier
- Conversion: 1 if there was a conversion in the 30 days after the impression; 0 otherwise
- Conversion ID: a unique identifier for each conversion
- Click: 1 if the impression was clicked; 0 otherwise
- Cost: the price paid for this ad
- Cat1-Cat9: categorical features associated with the ad. These features' semantic meaning is not disclosed.

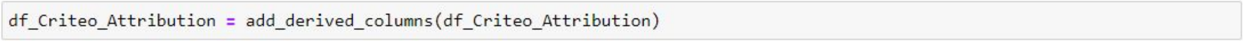
Data Preparation:

We start with the following initial transformation of the input data:

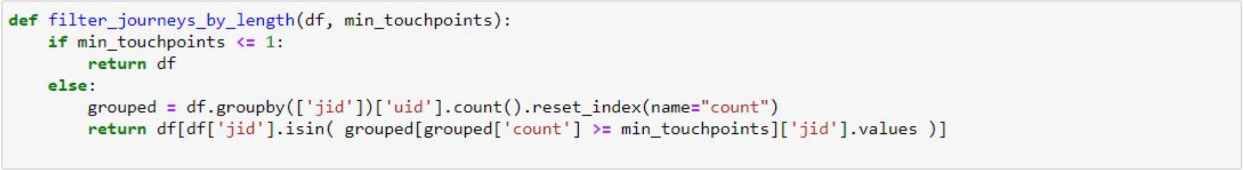
1. We aim to analyze entire customer journeys, i.e., sequences of events, so we introduce a journey ID (JID), which is a concatenation of the user ID and conversion ID.
2. We reduce the dataset size by randomly sampling 400 campaigns and filtering out journeys with just one event to focus on sequence analysis.
3. The original dataset is, of course, imbalanced because conversion events are very rare. We balance the dataset by downsampling non-converted journeys.
4. Finally, we also standardize some timestamp fields and do one-hot encoding for categorical fields (categories and campaigns). The total number of features after one-hot encoding is about 1,500.



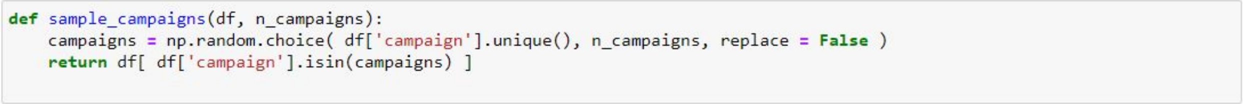
```
def add_derived_columns(df):
    df_ext = df.copy()
    df_ext['jid'] = df_ext['uid'].map(str) + '_' + df_ext['conversion_id'].map(str)

    min_max_scaler = MinMaxScaler()
    for cname in ('timestamp', 'time_since_last_click'):
        x = df_ext[cname].values.reshape(-1, 1)
        df_ext[cname + '_norm'] = min_max_scaler.fit_transform(x)

    return df_ext
```



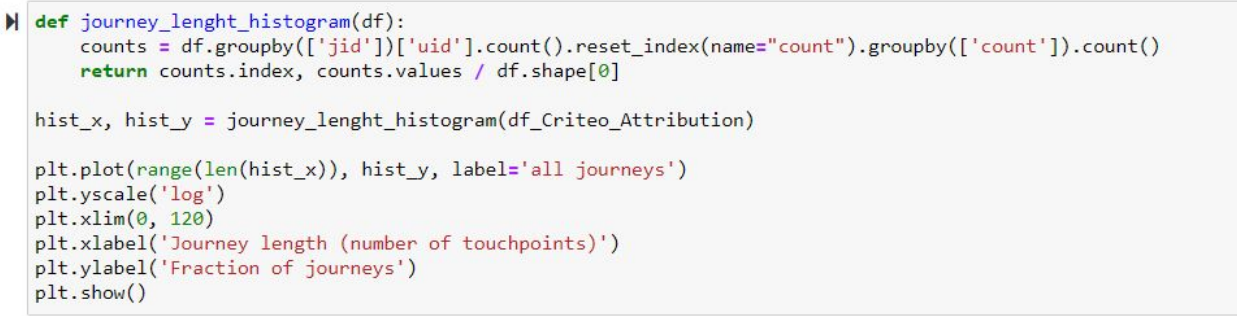
```
df_Criteo_Attribution = add_derived_columns(df_Criteo_Attribution)
```



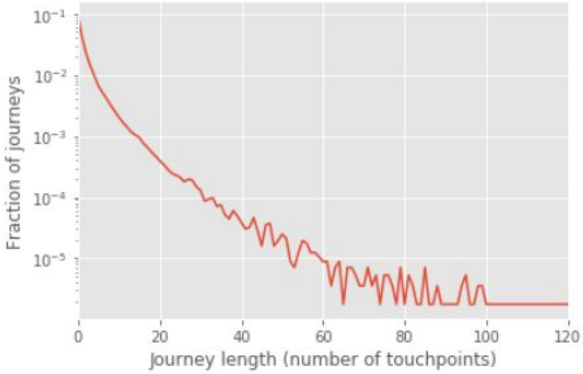
```
def filter_journeys_by_length(df, min_touchpoints):
    if min_touchpoints <= 1:
        return df
    else:
        grouped = df.groupby(['jid'])['uid'].count().reset_index(name="count")
        return df[df['jid'].isin( grouped[grouped['count'] >= min_touchpoints]['jid'].values )]
```



```
def sample_campaigns(df, n_campaigns):
    campaigns = np.random.choice( df['campaign'].unique(), n_campaigns, replace = False )
    return df[ df['campaign'].isin(campaigns) ]
```



```
def journey_lenght_histogram(df):
    counts = df.groupby(['jid'])['uid'].count().reset_index(name="count").groupby(['count']).count()
    return counts.index, counts.values / df.shape[0]

hist_x, hist_y = journey_lenght_histogram(df_Criteo_Attribution)

plt.plot(range(len(hist_x)), hist_y, label='all journeys')
plt.yscale('log')
plt.xlim(0, 120)
plt.xlabel('Journey length (number of touchpoints)')
plt.ylabel('Fraction of journeys')
plt.show()
```



Attribution Model:

An attribution model is the rule, or set of rules, that determines how conversion credit is assigned to different marketing touchpoints. Attribution models can be single-touch or multi-touch

Single-Touch Attribution Models

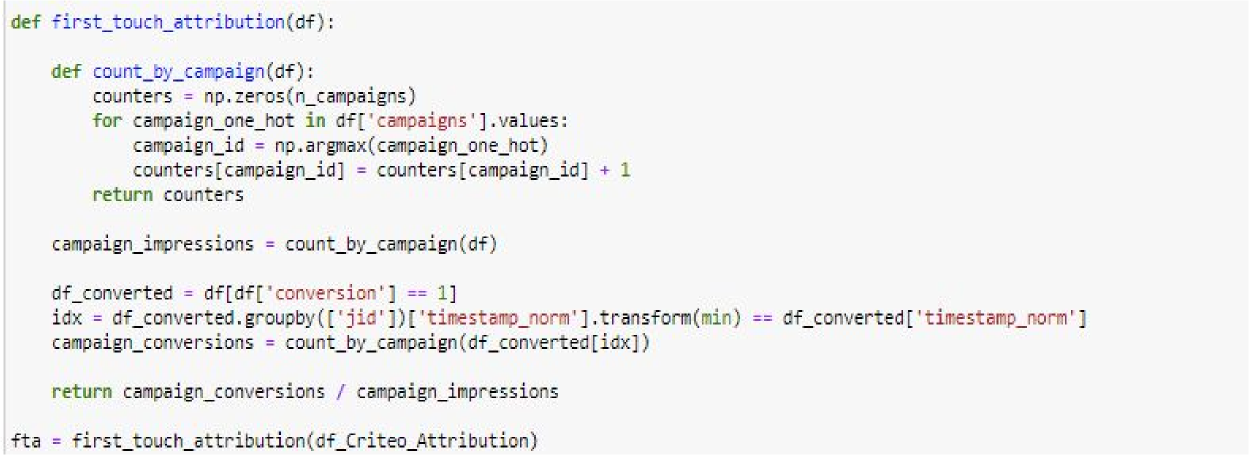
- Single-touch attribution models assign 100% of conversion credit to one marketing touchpoint. Even if a customer saw 20 ads before converting, single-touch attribution will determine that only 1 of the 20 ads deserves conversion credit.
- Single-touch attribution models are easy to implement because of their low level of complexity. They're also the most popular attribution models because of their historical tie to Google Ads
- The models are
  1. First Touch Attribution (FTA)
  2. Last Touch Attribution (LTA)

Multi-Touch Attribution Models:

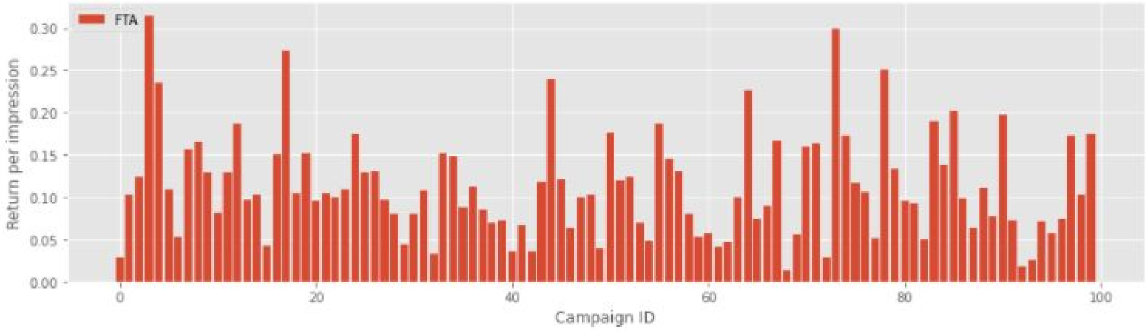
- Multi-touch attribution is best described in contrast to its counterpart, single-touch attribution.
- In order to divide the revenue credit for a sale properly, multi-touch attribution uses weighted modeling in order to allocate credit to the plethora of influential channels, campaigns, keywords, and touchpoints.
- Weighted touchpoint modeling assigns a percentage of the revenue credit for a customer to an array of touchpoints, as defined by the respective multi-touch attribution model chosen by the organization.
- The models are:
  1. Logistic Regression Model
  2. Time-decay Attribution Model
  3. Linear Attribution Model
  4. U-Shaped Attribution Model

First Touch Attribution:

- The First Touch model gives 100% of the credit to the marketing effort that drove a visitor to your website for the first time.
- Because it gives all the credit on the basis of a single touchpoint, it will naturally overemphasize a single part of the funnel. In this case, the First Touch model overemphasizes the top-of-the-funnel marketing channels that drive awareness

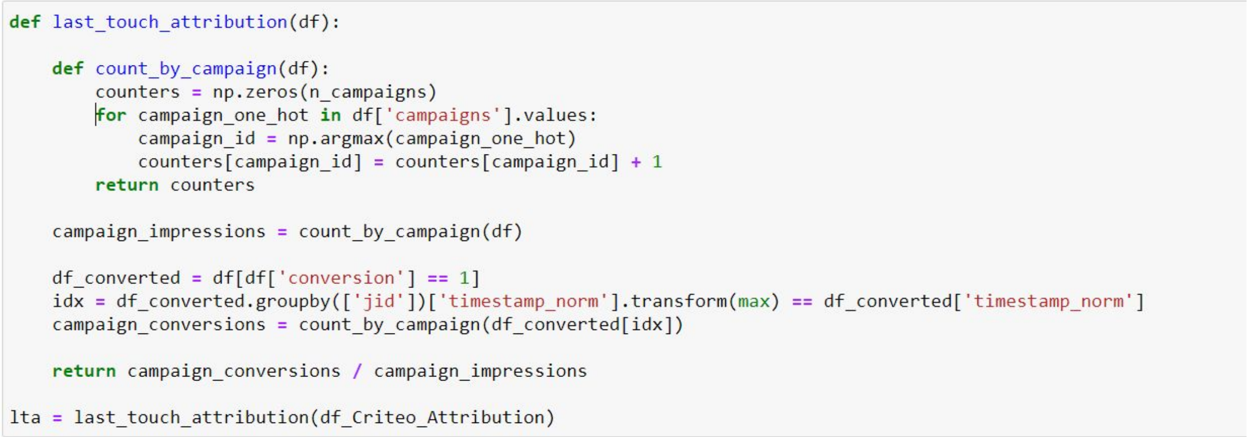


```
def first_touch_attribution(df):

    def count_by_campaign(df):
        counters = np.zeros(n_campaigns)
        for campaign_one_hot in df['campaigns'].values:
            campaign_id = np.argmax(campaign_one_hot)
            counters[campaign_id] = counters[campaign_id] + 1
        return counters

    campaign_impressions = count_by_campaign(df)

    df_converted = df[df['conversion'] == 1]
    idx = df_converted.groupby(['jid'])['timestamp_norm'].transform(min) == df_converted['timestamp_norm']
    campaign_conversions = count_by_campaign(df_converted[idx])

    return campaign_conversions / campaign_impressions

fta = first_touch_attribution(df_Criteo_Attribution)
```

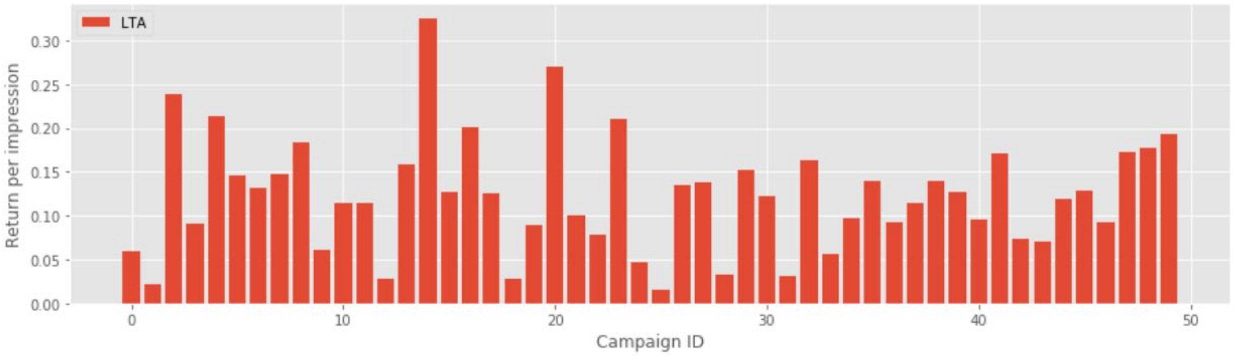


Last Touch Attribution:

- Last Touch Attribution model gives 100% of the credit for a conversion to the last click or visit that happened in a conversion path. If there was no click or visit, then it will credit the last impression.

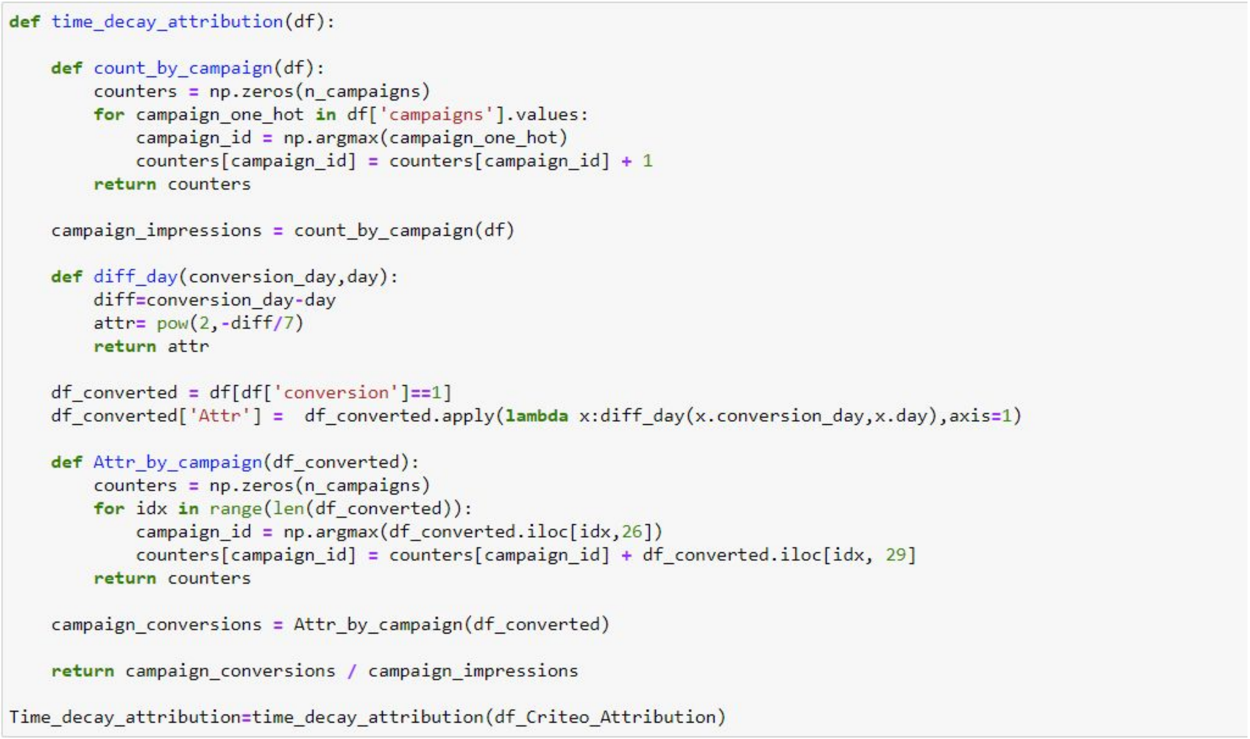


```
def last_touch_attribution(df):

    def count_by_campaign(df):
        counters = np.zeros(n_campaigns)
        for campaign_one_hot in df['campaigns'].values:
            campaign_id = np.argmax(campaign_one_hot)
            counters[campaign_id] = counters[campaign_id] + 1
        return counters

    campaign_impressions = count_by_campaign(df)

    df_converted = df[df['conversion'] == 1]
    idx = df_converted.groupby(['jid'])['timestamp_norm'].transform(max) == df_converted['timestamp_norm']
    campaign_conversions = count_by_campaign(df_converted[idx])

    return campaign_conversions / campaign_impressions

lta = last_touch_attribution(df_Criteo_Attribution)
```

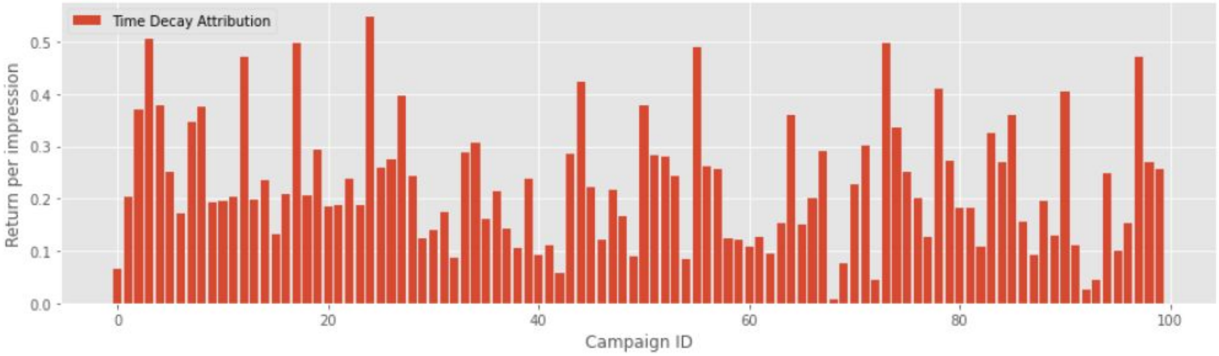


Time-decay attribution:

- The Time Decay model is a multi-touch model that gives more credit to the touchpoints closest to the conversion. It makes the assumption that the closer to the conversion, the more influence it had on the conversion.

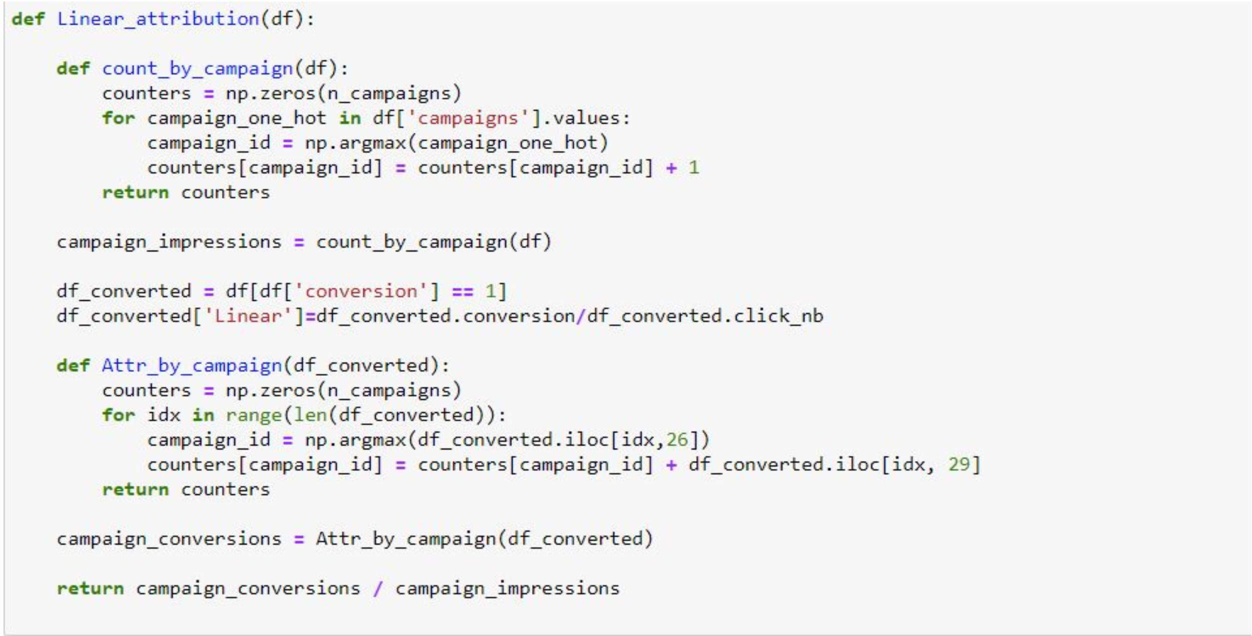


```
def time_decay_attribution(df):

    def count_by_campaign(df):
        counters = np.zeros(n_campaigns)
        for campaign_one_hot in df['campaigns'].values:
            campaign_id = np.argmax(campaign_one_hot)
            counters[campaign_id] = counters[campaign_id] + 1
        return counters

    campaign_impressions = count_by_campaign(df)

    def diff_day(conversion_day,day):
        diff=conversion_day-day
        attr= pow(2,-diff/7)
        return attr

    df_converted = df[df['conversion']==1]
    df_converted['Attr'] =  df_converted.apply(lambda x:diff_day(x.conversion_day,x.day),axis=1)

    def Attr_by_campaign(df_converted):
        counters = np.zeros(n_campaigns)
        for idx in range(len(df_converted)):
            campaign_id = np.argmax(df_converted.iloc[idx,26])
            counters[campaign_id] = counters[campaign_id] + df_converted.iloc[idx, 29]
        return counters

    campaign_conversions = Attr_by_campaign(df_converted)

    return campaign_conversions / campaign_impressions

Time_decay_attribution=time_decay_attribution(df_Criteo_Attribution)
```

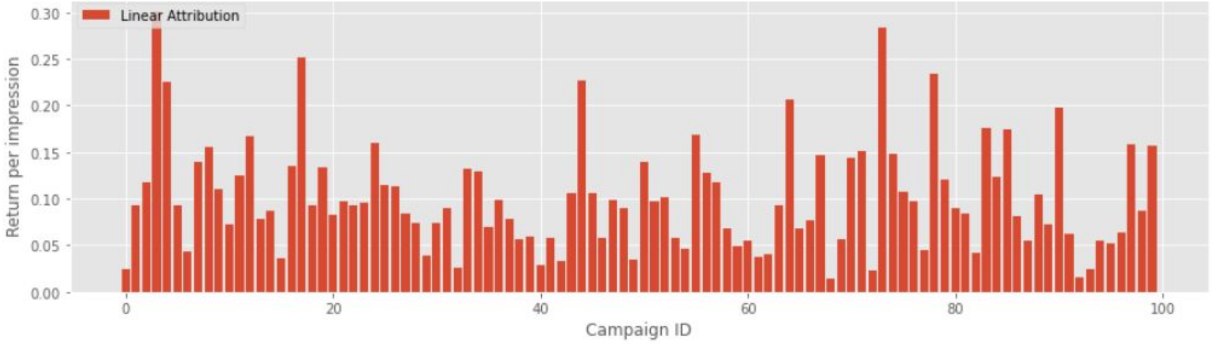


Linear Attribution:

- Linear is the simplest of the multi-touch attribution models. It distributes credit by evenly applying credit to every single touch in the buyer journey.
- The positive is that it is a multi-touch model, so it gives credit to marketing channels throughout the multiple stages of the funnel.
- The negative is that it doesn't take into account the potential for varying impact of marketing touches

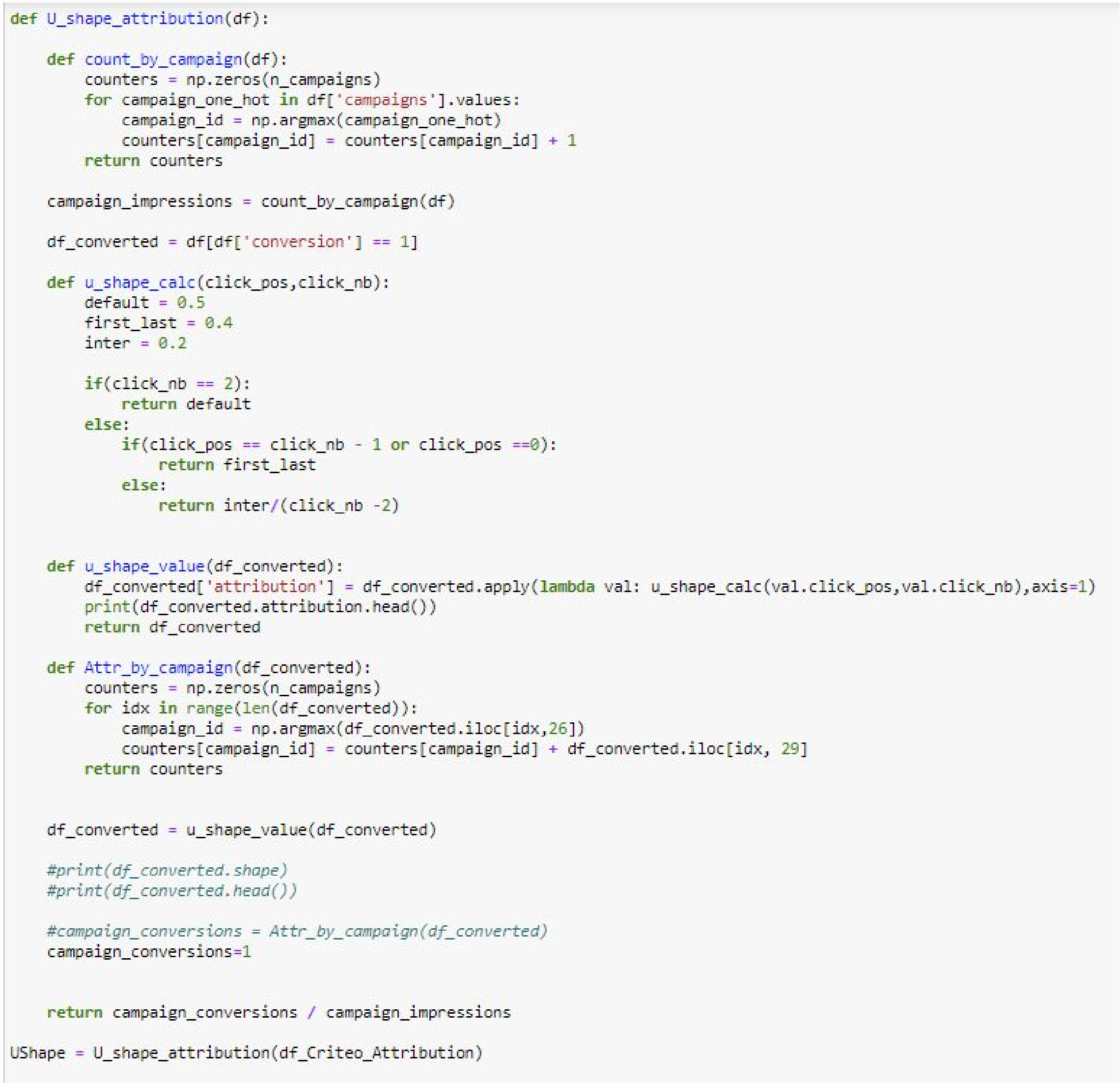


```
def Linear_attribution(df):

    def count_by_campaign(df):
        counters = np.zeros(n_campaigns)
        for campaign_one_hot in df['campaigns'].values:
            campaign_id = np.argmax(campaign_one_hot)
            counters[campaign_id] = counters[campaign_id] + 1
        return counters

    campaign_impressions = count_by_campaign(df)

    df_converted = df[df['conversion'] == 1]
    df_converted['Linear']=df_converted.conversion/df_converted.click_nb

    def Attr_by_campaign(df_converted):
        counters = np.zeros(n_campaigns)
        for idx in range(len(df_converted)):
            campaign_id = np.argmax(df_converted.iloc[idx,26])
            counters[campaign_id] = counters[campaign_id] + df_converted.iloc[idx, 29]
        return counters

    campaign_conversions = Attr_by_campaign(df_converted)

    return campaign_conversions / campaign_impressions
```

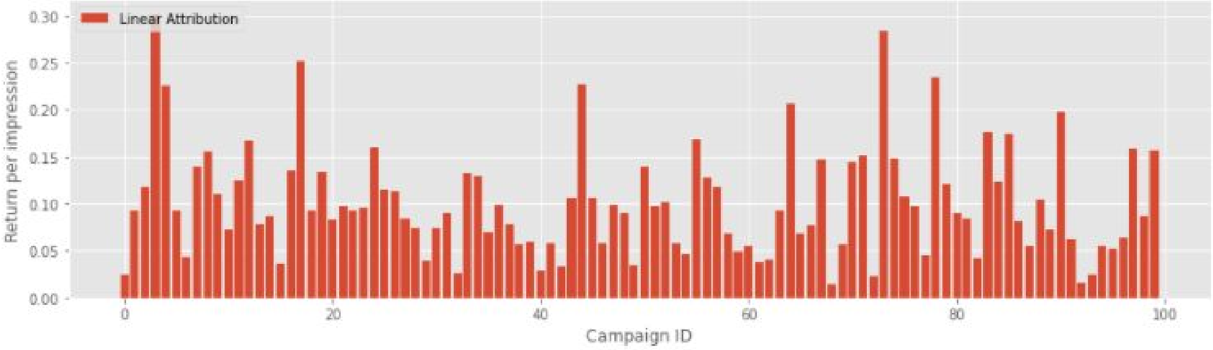


U-shaped attribution:

- The U-Shaped model, which Google calls Position-Based, is a great multi-touch attribution model for marketing teams that focus on lead generation.
- It’s a multi-touch model that tracks every single touchpoint, but rather than give equal credit to all touchpoints like the Linear model, it emphasizes the importance of two key touchpoints:
    1. The anonymous first touch that got the visitor in the door
    2. The lead conversion touch.
- These two touches get 40% credit each and the remaining touchpoints equally split the remaining 20%

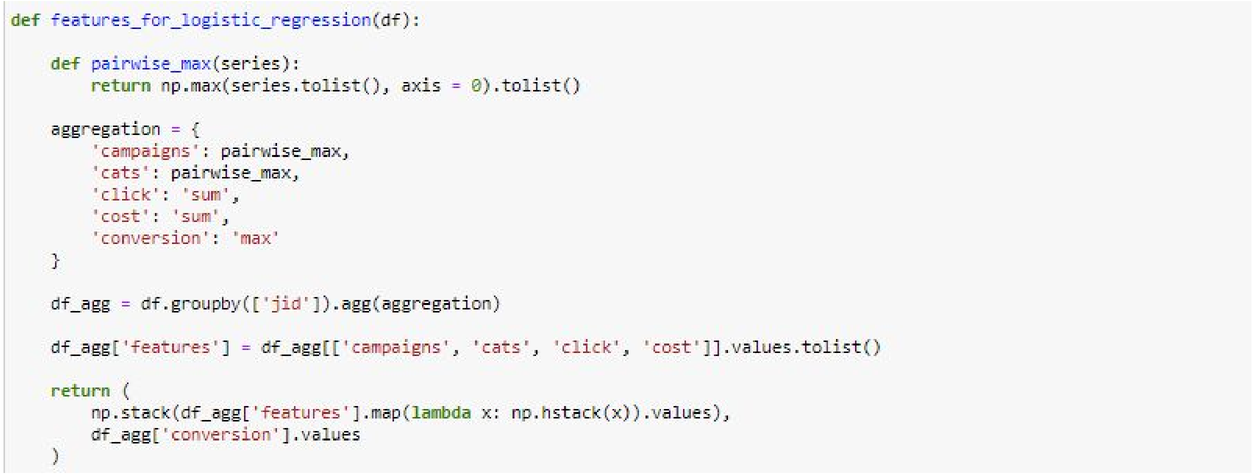


```
def U_shape_attribution(df):

    def count_by_campaign(df):
        counters = np.zeros(n_campaigns)
        for campaign_one_hot in df['campaigns'].values:
            campaign_id = np.argmax(campaign_one_hot)
            counters[campaign_id] = counters[campaign_id] + 1
        return counters

    campaign_impressions = count_by_campaign(df)

    df_converted = df[df['conversion'] == 1]

    def u_shape_calc(click_pos,click_nb):
        default = 0.5
        first_last = 0.4
        inter = 0.2

        if(click_nb == 2):
            return default
        else:
            if(click_pos == click_nb - 1 or click_pos ==0):
                return first_last
            else:
                return inter/(click_nb -2)


    def u_shape_value(df_converted):
        df_converted['attribution'] = df_converted.apply(lambda val: u_shape_calc(val.click_pos,val.click_nb),axis=1)
        print(df_converted.attribution.head())
        return df_converted

    def Attr_by_campaign(df_converted):
        counters = np.zeros(n_campaigns)
        for idx in range(len(df_converted)):
            campaign_id = np.argmax(df_converted.iloc[idx,26])
            counters[campaign_id] = counters[campaign_id] + df_converted.iloc[idx, 29]
        return counters


    df_converted = u_shape_value(df_converted)

    #print(df_converted.shape)
    #print(df_converted.head())

    #campaign_conversions = Attr_by_campaign(df_converted)
    campaign_conversions=1


    return campaign_conversions / campaign_impressions

UShape = U_shape_attribution(df_Criteo_Attribution)
```

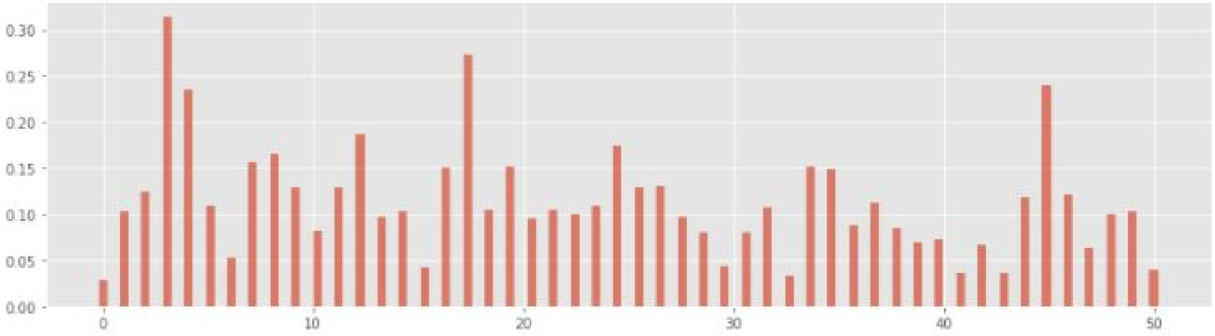


Logistic Regression Model:

- Logistic Regression Analysis aims to reveal touchpoints true contributions.
- The idea of a regression-based approach is straightforward: Each journey is represented as a vector in which each campaign is represented by a binary feature (and can be other event features), a regression model is fit to predict conversions, and the resulting regression coefficients are interpreted as attribution weights.



```
def features_for_logistic_regression(df):

    def pairwise_max(series):
        return np.max(series.tolist(), axis = 0).tolist()

    aggregation = {
        'campaigns': pairwise_max,
        'cats': pairwise_max,
        'click': 'sum',
        'cost': 'sum',
        'conversion': 'max'
    }

    df_agg = df.groupby(['jid']).agg(aggregation)

    df_agg['features'] = df_agg[['campaigns', 'cats', 'click', 'cost']].values.tolist()

    return (
        np.stack(df_agg['features'].map(lambda x: np.hstack(x)).values),
        df_agg['conversion'].values
    )
```

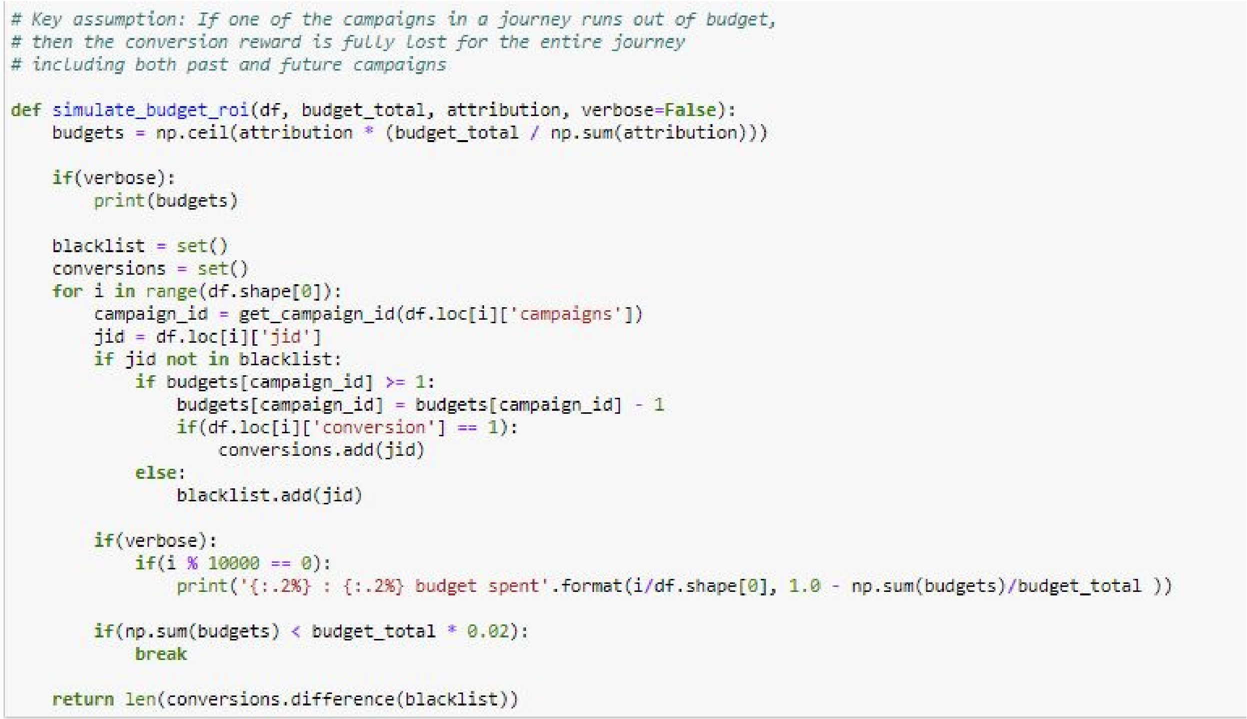


Budget Optimization:

The campaign simulation idea can be outlined as follows:

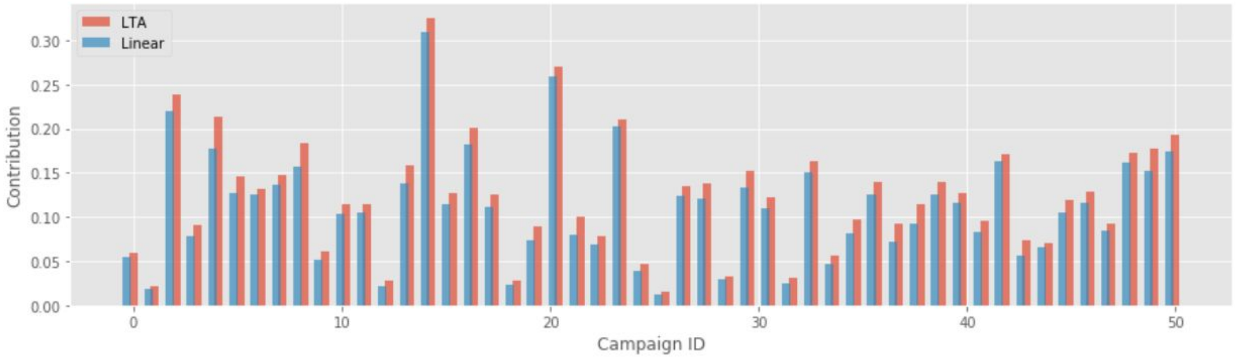
1. At the beginning of the process, we distribute a limited budget across the campaigns according to the attribution weights.

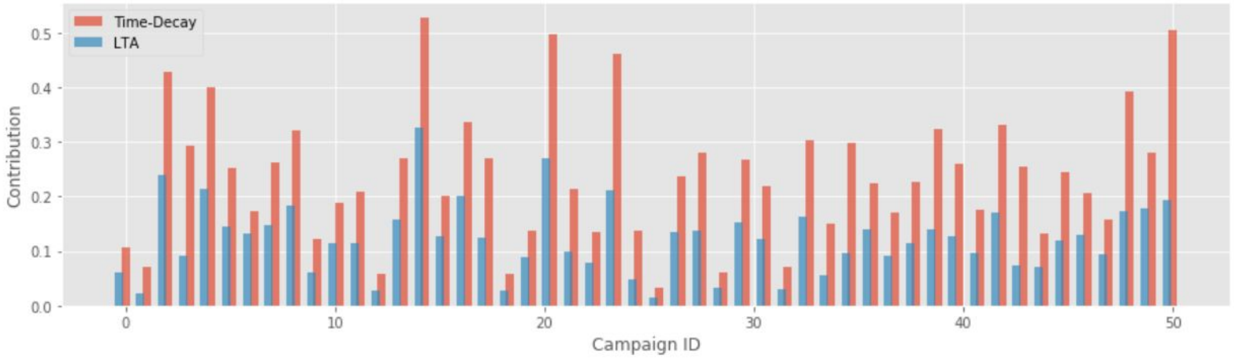
2. We replay the available historical events (ordered by their timestamps) and decrement the budgets accordingly.
3. Once a campaign runs out of money, we stop to replay the remaining events associated with it and somehow estimate the probabilities of conversion for all journeys affected by this campaign's suppression.
4. Finally, we count the total number of conversions and estimate ROI. If none of the campaigns in a converted journey runs out of money before the journey ends, this conversion will be counted explicitly. Otherwise, the estimate of the conversion probability will be used.

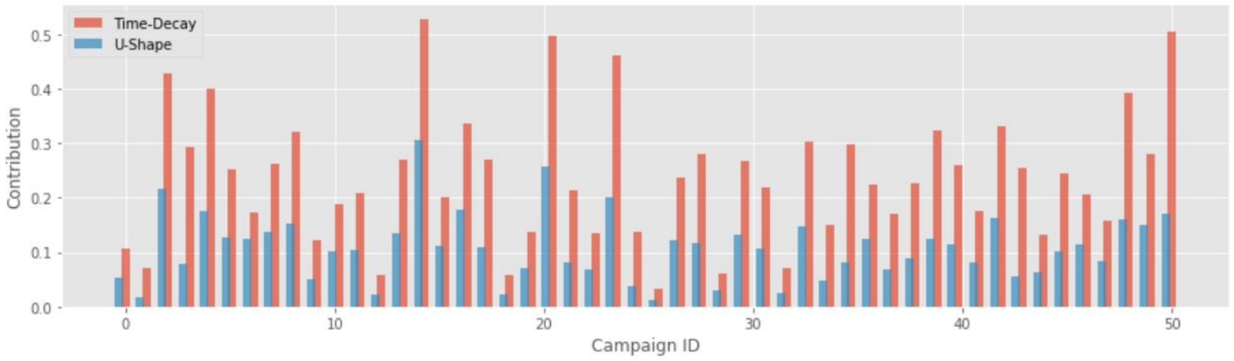


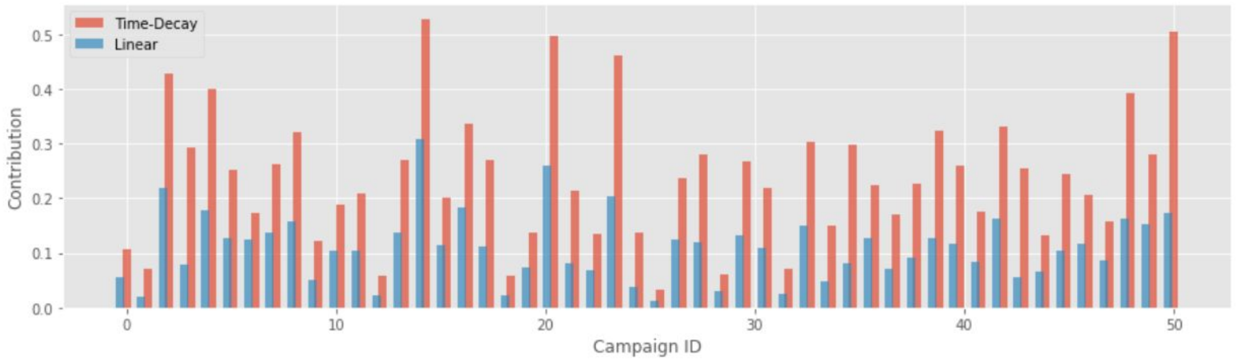
```python
# Key assumption: If one of the campaigns in a journey runs out of budget,
# then the conversion reward is fully lost for the entire journey
# including both past and future campaigns

def simulate_budget_roi(df, budget_total, attribution, verbose=False):
    budgets = np.ceil(attribution * (budget_total / np.sum(attribution)))

    if(verbose):
        print(budgets)

    blacklist = set()
    conversions = set()
    for i in range(df.shape[0]):
        campaign_id = get_campaign_id(df.loc[i]['campaigns'])
        jid = df.loc[i]['jid']
        if jid not in blacklist:
            if budgets[campaign_id] >= 1:
                budgets[campaign_id] = budgets[campaign_id] - 1
                if(df.loc[i]['conversion'] == 1):
                    conversions.add(jid)
            else:
                blacklist.add(jid)

        if(verbose):
            if(i % 10000 == 0):
                print('{:.2%} : {:.2%} budget spent'.format(i/df.shape[0], 1.0 - np.sum(budgets)/budget_total ))

        if(np.sum(budgets) < budget_total * 0.02):
            break

    return len(conversions.difference(blacklist))
```

Comparison of Models:



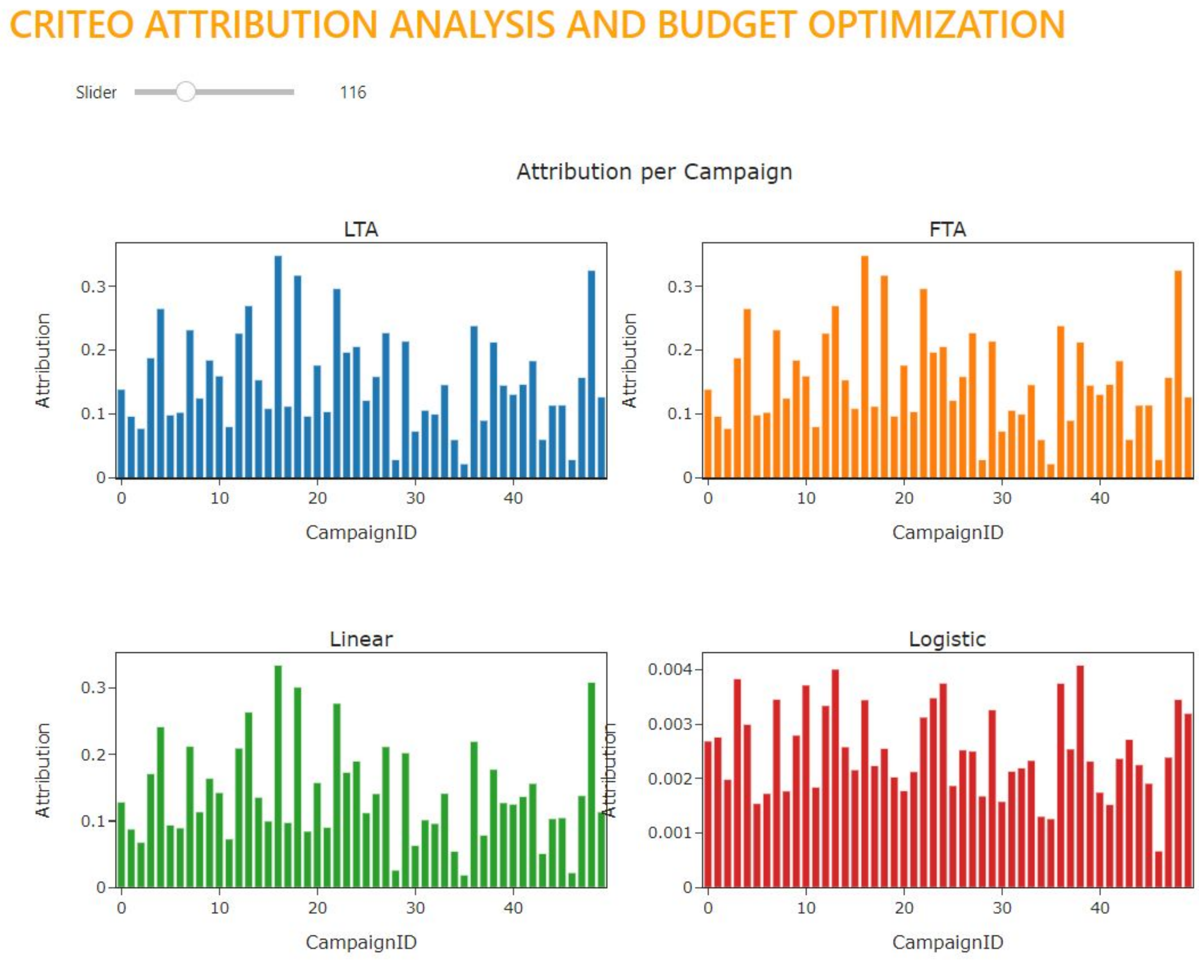






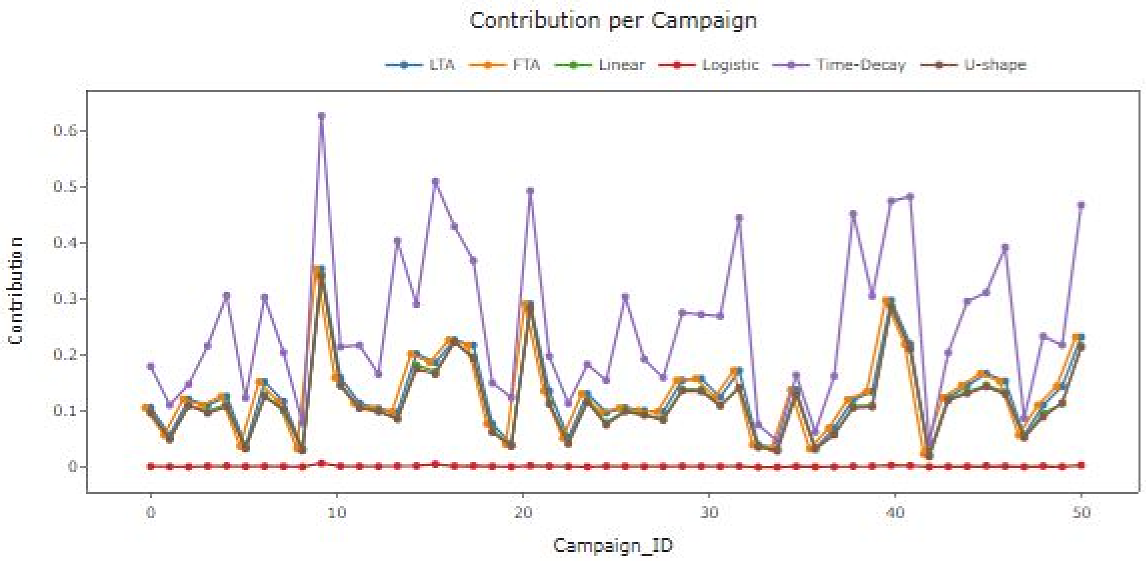
Voila Dashboard:

- Voilà allows you to convert a Jupyter Notebook into an interactive dashboard that allows you to share your work with others. It is secure and customizable, giving you control over what your readers experience.
- Voilà turns Jupyter notebooks into standalone applications. Use these examples for inspiration when creating your own dashboard applications.

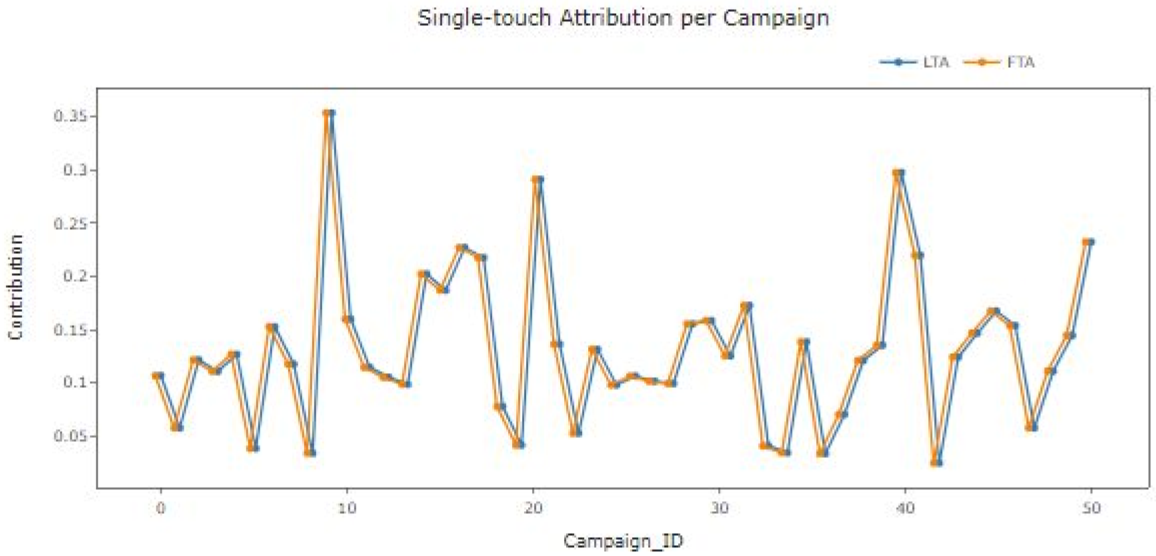


CRITEO ATTRIBUTION ANALYSIS AND BUDGET OPTIMIZATION

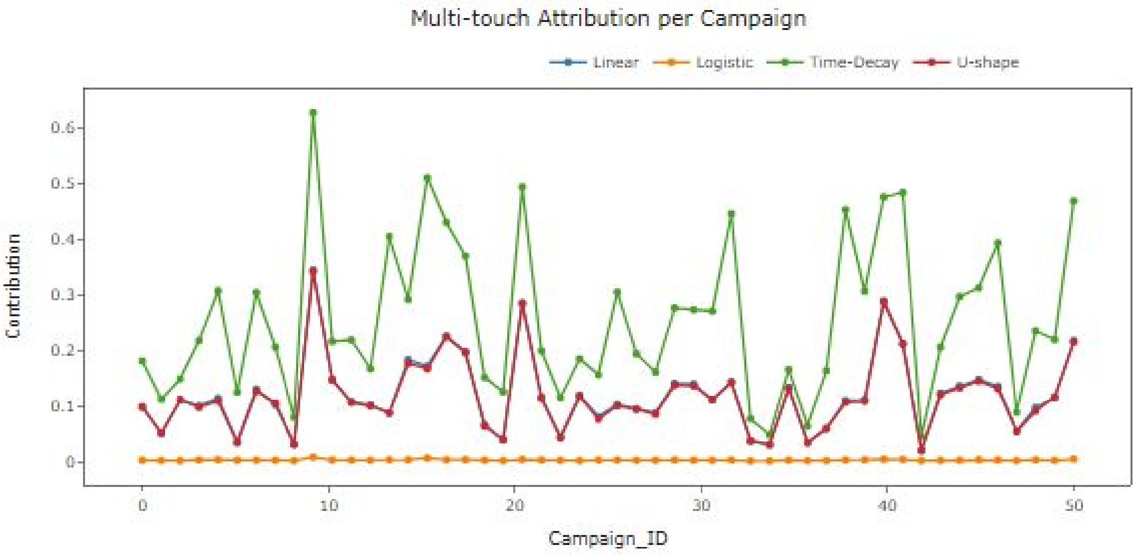
Slider 116

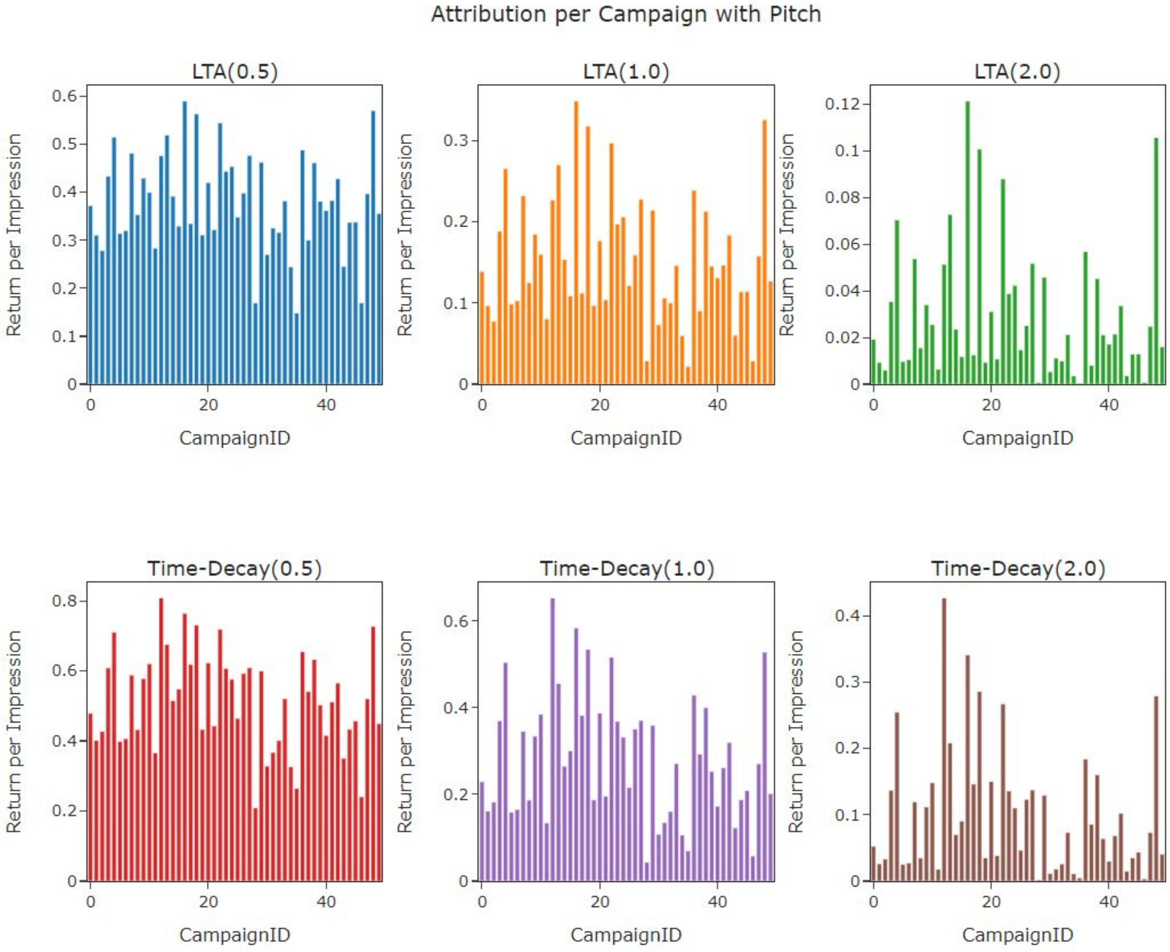


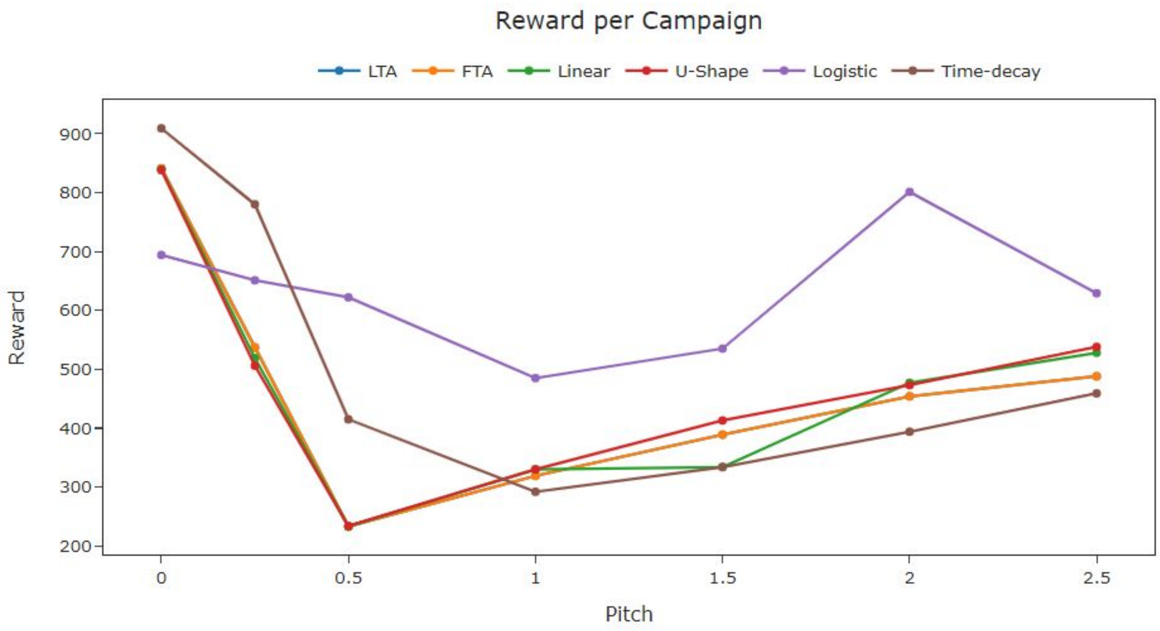
Contribution per Campaign

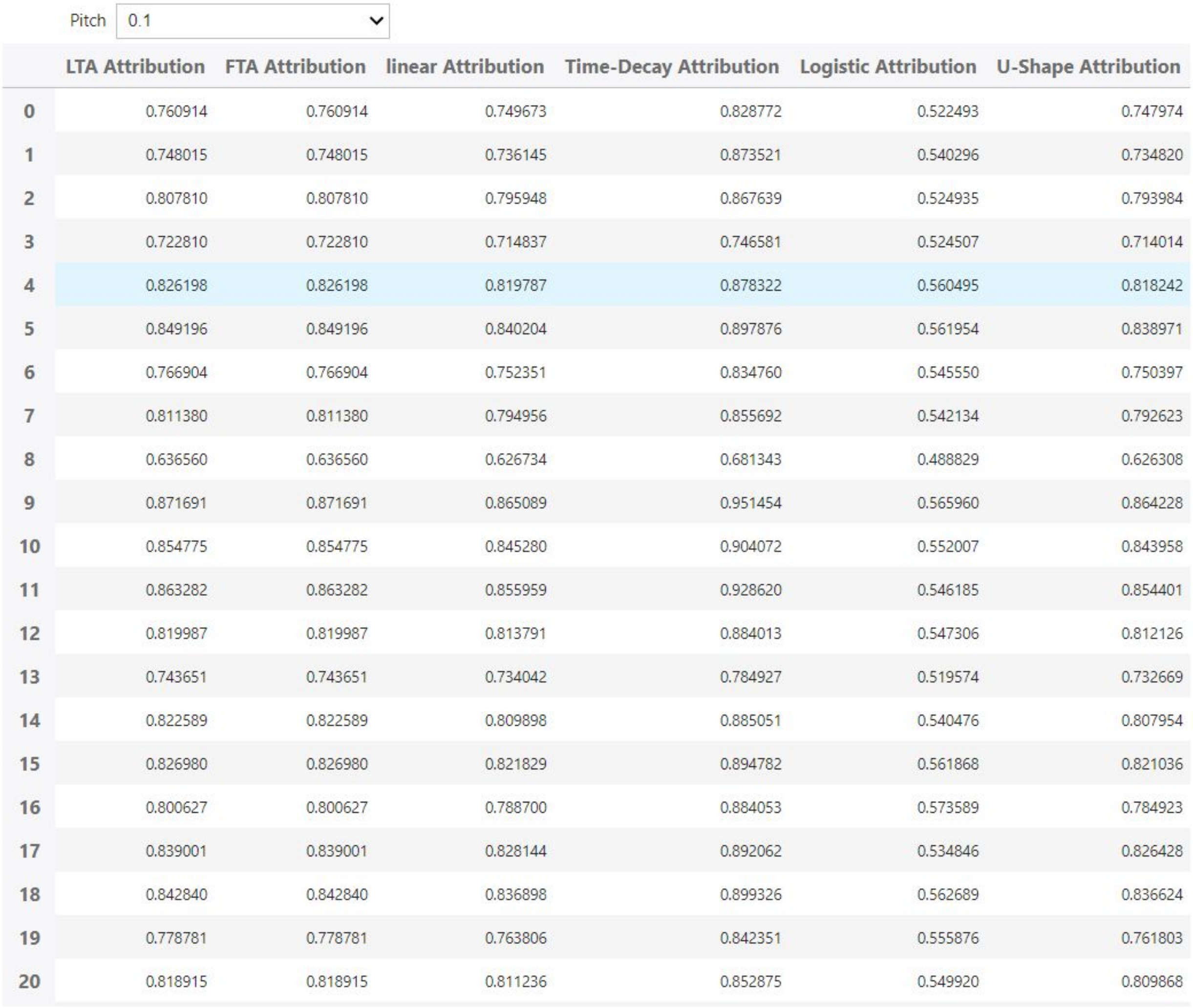


Single-touch Attribution per Campaign


Multi-touch Attribution per Campaign
Linear
Logistic
Time-Decay
U-shape
Contribution
0.6
0.5
0.4
0.3
0.2
0.1
0
0
10
20
30
40
50
Campaign_ID


Attribution per Campaign with Pitch
LTA(0.5)
LTA(1.0)
LTA(2.0)
Time-Decay(0.5)
Time-Decay(1.0)
Time-Decay(2.0)
Return per Impression
0.6
0.5
0.4
0.3
0.2
0.1
0
0.3
0.2
0.1
0
0.12
0.1
0.08
0.06
0.04
0.02
0
0.8
0.6
0.4
0.2
0
0.6
0.4
0.2
0
0.4
0.3
0.2
0.1
0
0
20
40
CampaignID

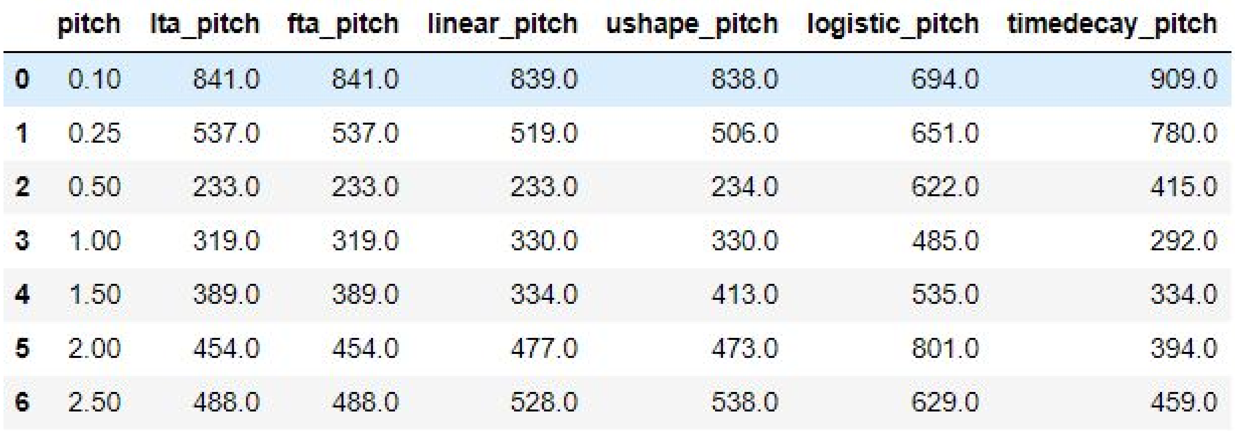




Pitch 0.1

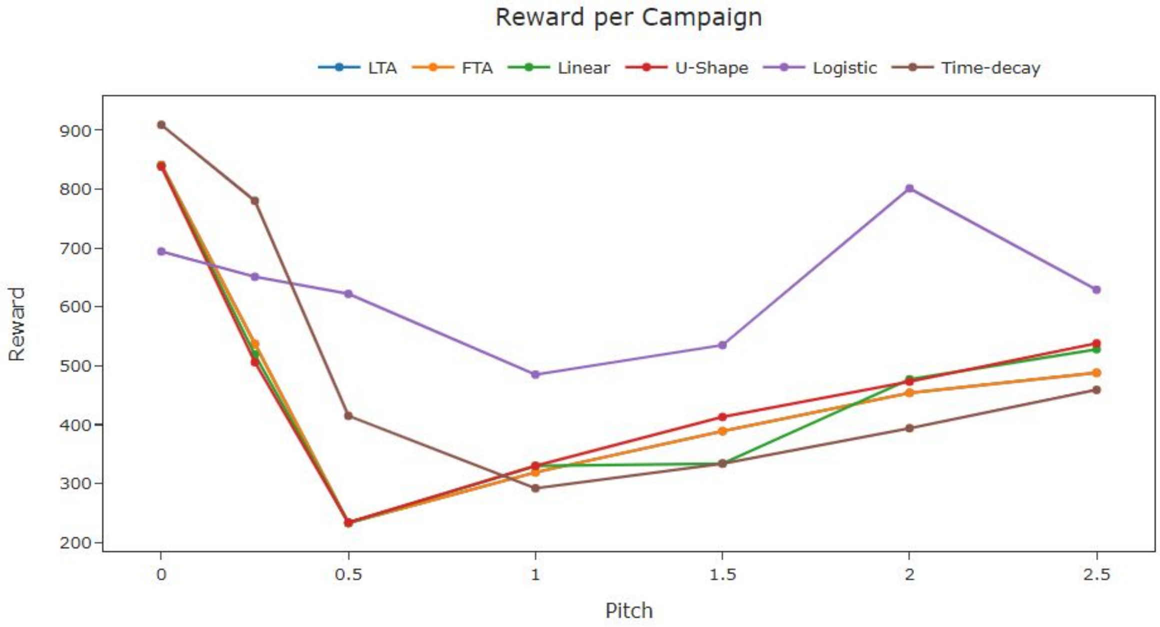
	LTA Attribution	FTA Attribution	linear Attribution	Time-Decay Attribution	Logistic Attribution	U-Shape Attribution
0	0.760914	0.760914	0.749673	0.828772	0.522493	0.747974
1	0.748015	0.748015	0.736145	0.873521	0.540296	0.734820
2	0.807810	0.807810	0.795948	0.867639	0.524935	0.793984
3	0.722810	0.722810	0.714837	0.746581	0.524507	0.714014
4	0.826198	0.826198	0.819787	0.878322	0.560495	0.818242
5	0.849196	0.849196	0.840204	0.897876	0.561954	0.838971
6	0.766904	0.766904	0.752351	0.834760	0.545550	0.750397
7	0.811380	0.811380	0.794956	0.855692	0.542134	0.792623
8	0.636560	0.636560	0.626734	0.681343	0.488829	0.626308
9	0.871691	0.871691	0.865089	0.951454	0.565960	0.864228
10	0.854775	0.854775	0.845280	0.904072	0.552007	0.843958
11	0.863282	0.863282	0.855959	0.928620	0.546185	0.854401
12	0.819987	0.819987	0.813791	0.884013	0.547306	0.812126
13	0.743651	0.743651	0.734042	0.784927	0.519574	0.732669
14	0.822589	0.822589	0.809898	0.885051	0.540476	0.807954
15	0.826980	0.826980	0.821829	0.894782	0.561868	0.821036
16	0.800627	0.800627	0.788700	0.884053	0.573589	0.784923
17	0.839001	0.839001	0.828144	0.892062	0.534846	0.826428
18	0.842840	0.842840	0.836898	0.899326	0.562689	0.836624
19	0.778781	0.778781	0.763806	0.842351	0.555876	0.761803
20	0.818915	0.818915	0.811236	0.852875	0.549920	0.809868

Recommended Model:



	pitch	lta_pitch	fta_pitch	linear_pitch	ushape_pitch	logistic_pitch	timedecay_pitch
0	0.10	841.0	841.0	839.0	838.0	694.0	909.0
1	0.25	537.0	537.0	519.0	506.0	651.0	780.0
2	0.50	233.0	233.0	233.0	234.0	622.0	415.0
3	1.00	319.0	319.0	330.0	330.0	485.0	292.0
4	1.50	389.0	389.0	334.0	413.0	535.0	334.0
5	2.00	454.0	454.0	477.0	473.0	801.0	394.0
6	2.50	488.0	488.0	528.0	538.0	629.0	459.0

- When we compare the reward for each model, we can see that Time-decay Attribution Model is most ideal when the pitch = 0.1
- When we look at the graph, the Logistic Regression model has higher reward than compared to all the other models at most of the pitches


Reward per Campaign
LTA
FTA
Linear
U-Shape
Logistic
Time-decay
Reward
Pitch