Attribution Models

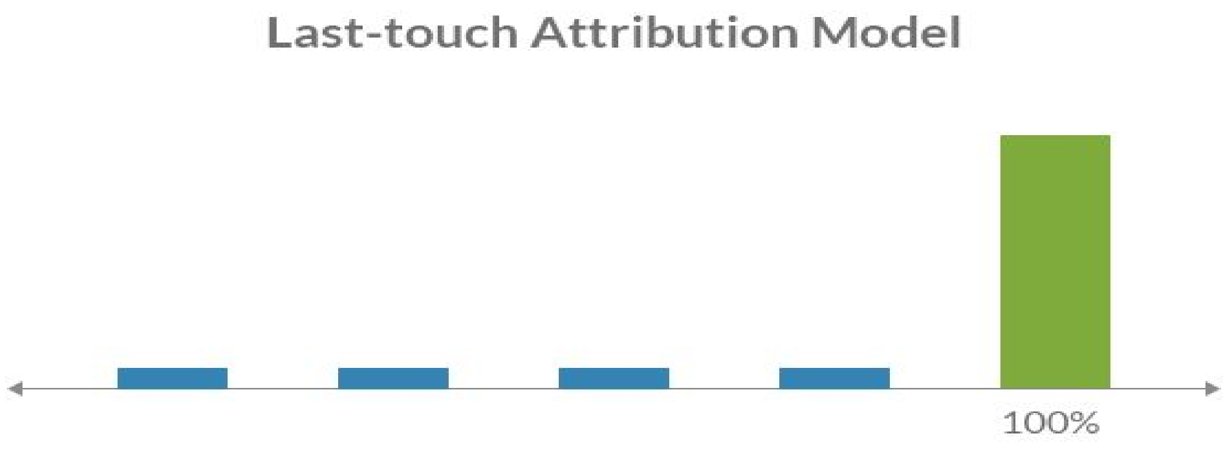
To incorporate the various Attribution Modeling and Budget Optimization techniques and to choose the best model amongst the various models based on the ROI generated.

Models:

- Last Touch Attribution (LTA)
- First Touch Attribution (FTA)
- Linear Attribution
- Time Decay Attribution
- U-Shaped/Position Based Attribution

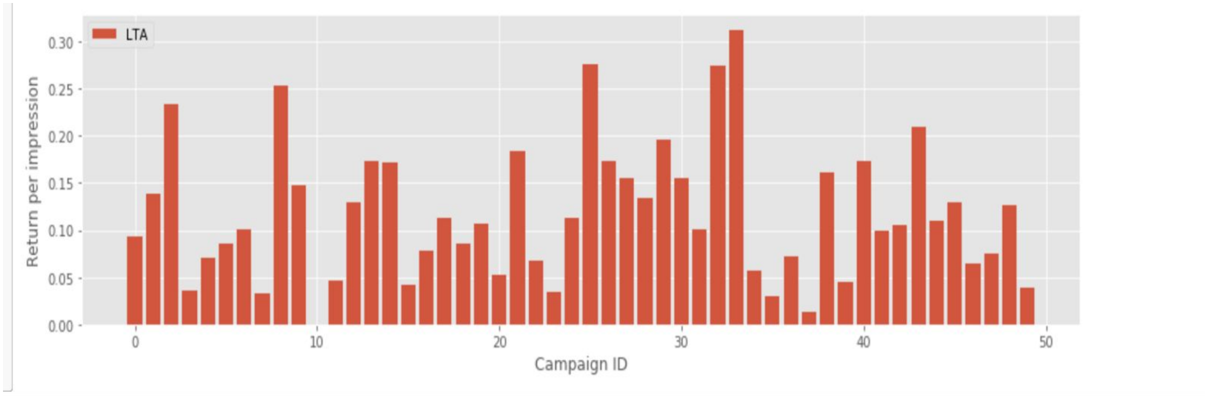
Last Touch Attribution (LTA):

Last touch attribution is the last of the single touchpoint attribution models. The entire credit is assigned to the last marketing touchpoint.



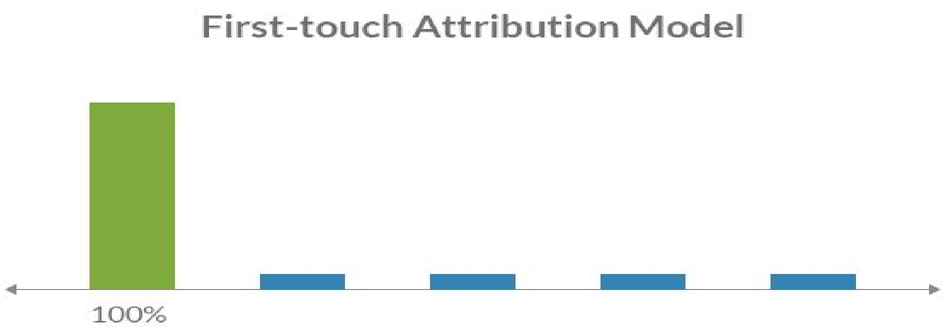
```python
def last_touch_attribution(df):

    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(df6)
```

Results:



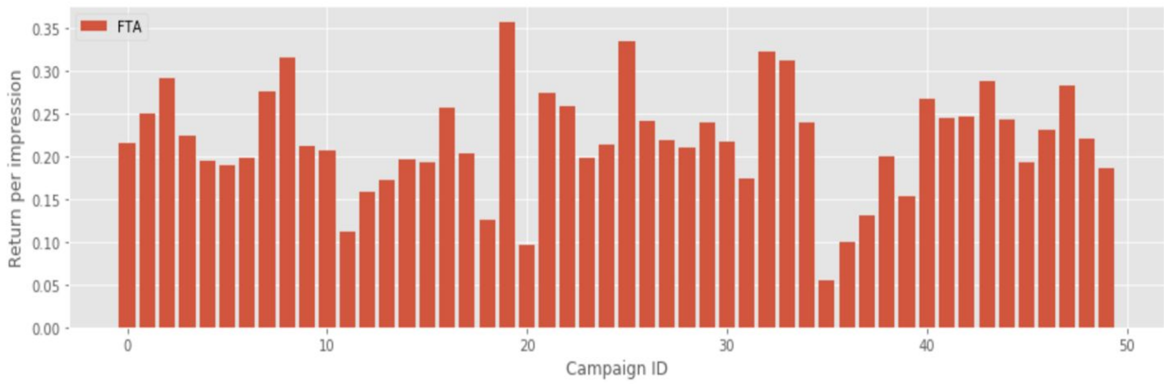
First Touch Attribution (FTA) :

First touch attribution is the first of the single touchpoint attribution models. The entire credit is assigned to the first marketing touchpoint.



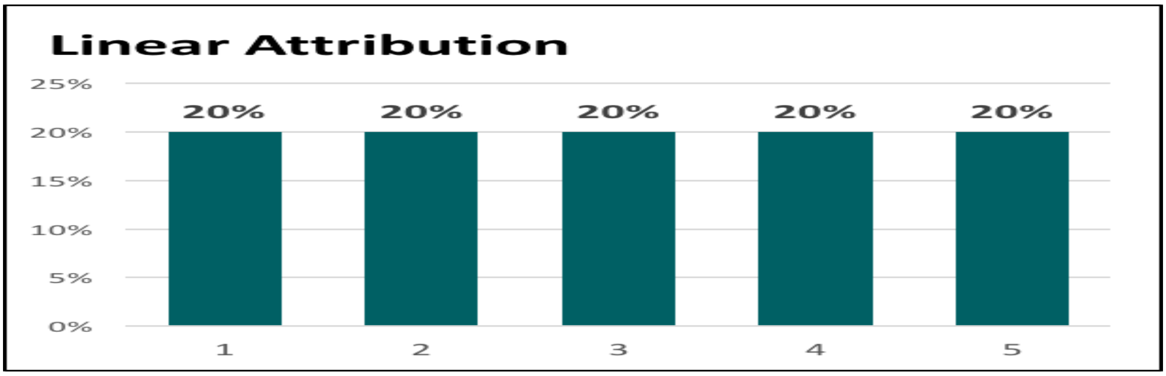
```
def first_touch_attribution(df):

    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(['uid'])['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(df6)
```

Results:



Linear Attribution Model:

In linear attribution model, for a conversion, the credit is split equally between all the interactions the customer had with the business.

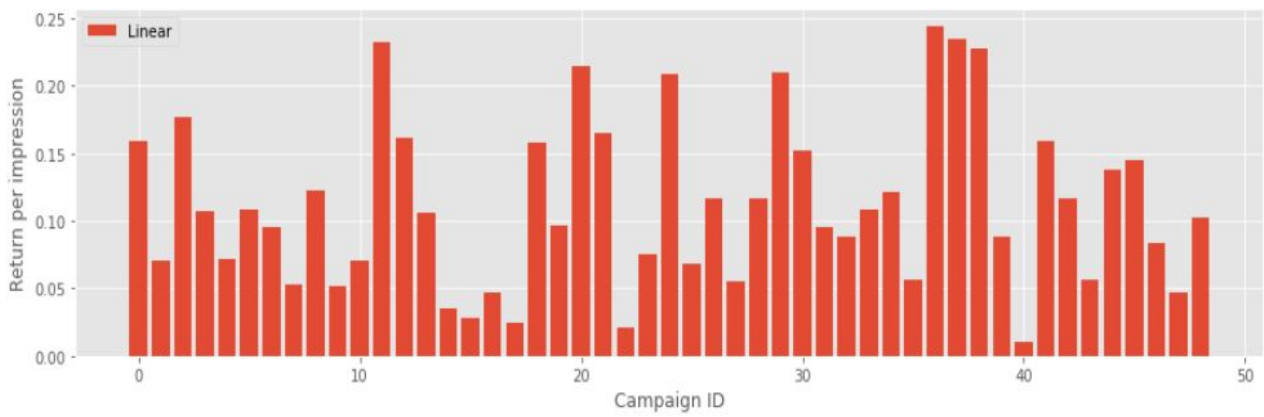



```
for index , row in df_merge.iterrows():
  camp_prob.append(1/row['campaign_count'])
df_merge['campaign_probability'] =    camp_prob
```

```
camp_df.head()
```

	campaign	campaign_probability	campaign_count	campaign_wt
0	73325	42.458333	433	0.098056
1	73328	128.196769	805	0.159251
2	83677	67.793407	961	0.070545
3	336258	11.000000	62	0.177419
4	442617	34.119048	319	0.106956

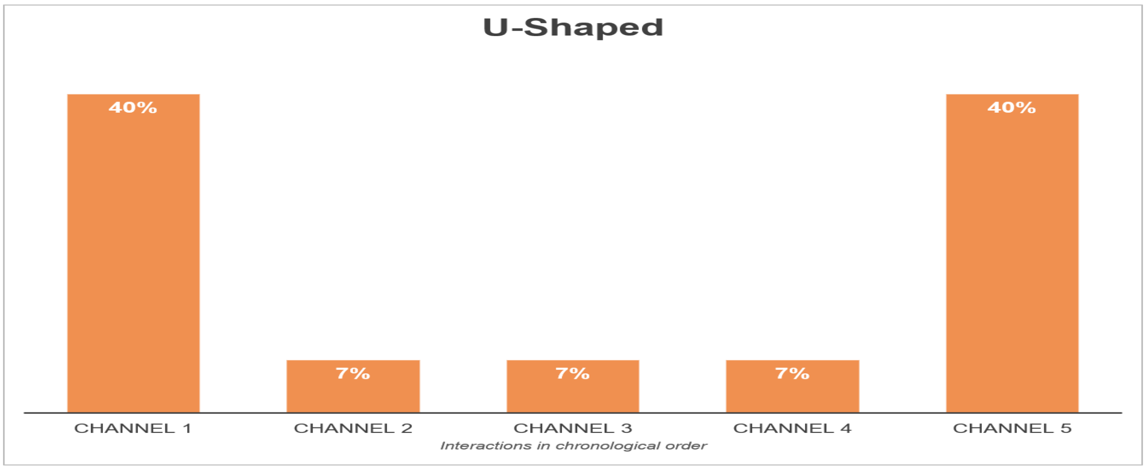
Result:



U-Shaped/Position Based Attribution Model:

The Position Based attribution model (also called U-shaped attribution) splits the credit for a sale between a prospect's first interaction with your brand and the moment they convert to a lead.

40% of the credit is given to each of these points, with the remaining 20% spread out between any other interactions that happened in the middle.



```python
time_min = df_conv_user.groupby(['uid'])['campaign','timestamp_norm'].min()
time_max = df_conv_user.groupby(['uid'])['campaign','timestamp_norm'].max()
```

```python
df_min_time = pd.DataFrame(time_min)
df_max_time = pd.DataFrame(time_max)
```

```python
df_min_time = df_min_time.reset_index()
df_max_time = df_max_time.reset_index()
```

```python
df_min_time
df_min_time = df_min_time.rename(columns={"timestamp_norm":"min_timestamp"})
```

```python
df_max_time.head()
df_max_time = df_max_time.rename(columns={"timestamp_norm":"max_timestamp"})
```

```python
df_merge_time_max = pd.merge(df_merge,df_max_time,  how = 'inner', on =(['uid','campaign']),)
df_merge_time_min = pd.merge(df_merge_time_max,df_min_time,  how = 'inner', on =(['uid','campaign']),)
```

```python
df_u_shape = df_merge_time_min
```

```python
df_u_shape = df_u_shape.drop(columns=['campaign_probability'])
```

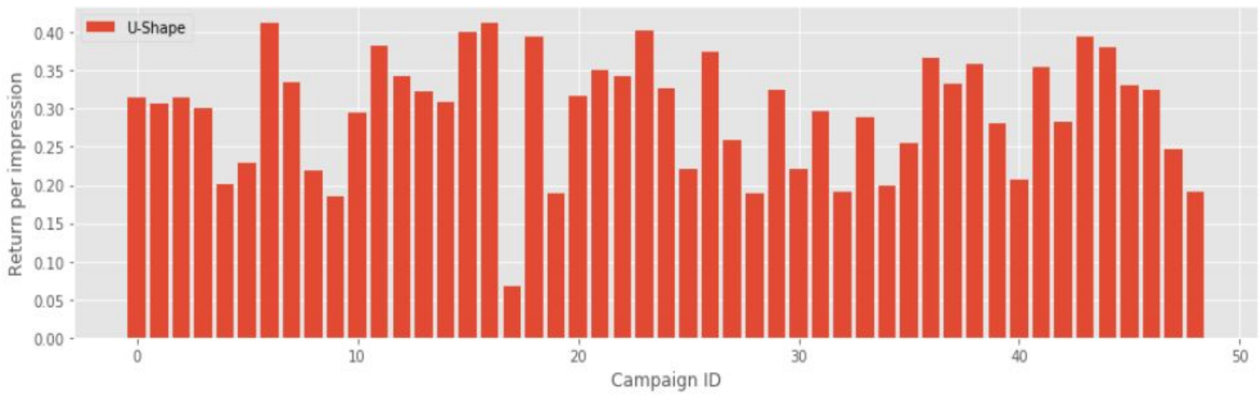
```python
df_u_shape['u_prob'] = np.where(df_u_shape['timestamp_norm']==df_u_shape['max_timestamp'],
                                '0.4',
                                np.where(df_u_shape['timestamp_norm']==df_u_shape['min_timestamp'],
                                '0.4', 0.2*(1/(df_u_shape['campaign_count']-2))))
```

```python
df_u_shape.loc[df_u_shape.campaign_count == 1, 'u_prob'] = 1
df_u_shape.loc[df_u_shape.campaign_count == 2, 'u_prob'] = 0.5
```

	campaign	u_prob	campaign_count	campaign_wt
0	73325	41.000000	173	0.236994
1	73328	129.263054	411	0.314509
2	83677	68.636364	224	0.306412
3	336258	11.000000	35	0.314286
4	442617	34.600000	115	0.300870
...	...	...	...	...
395	32385772	17.000000	58	0.293103
396	32398755	90.223816	457	0.197426
397	32398758	58.857143	296	0.198842
398	32405311	26.000000	96	0.270833
399	32452108	34.756190	128	0.271533

400 rows × 4 columns

Results:



Time Decay Attribution Model:

Time Decay attribution is similar to Linear attribution - it spreads out the value across multiple events. But unlike Linear attribution, the Time Decay model also takes into consideration when the touchpoint occurred.

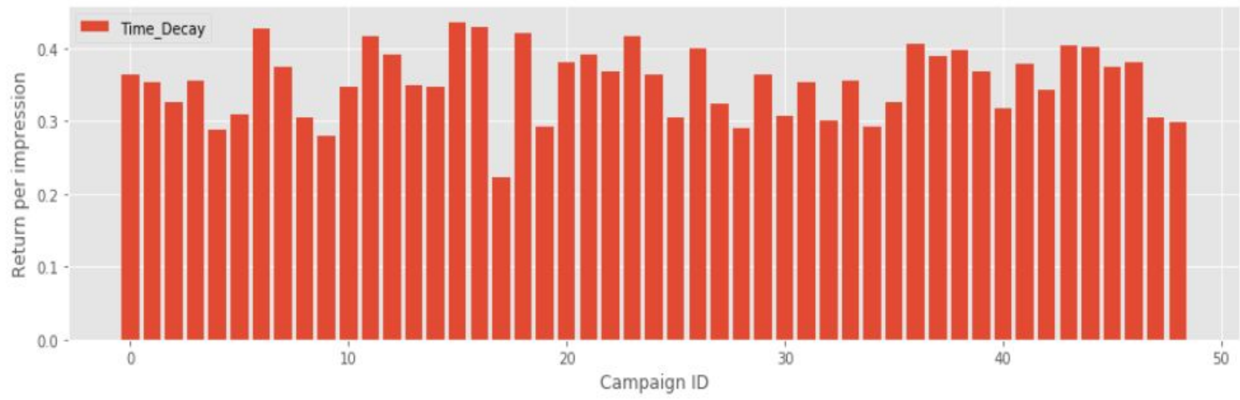
Interactions that occur closer to the time of purchase have more value attributed to them. The first interaction gets less credit, while the last interaction will get the most.

```python
df_time_decay['time_decay_prob'] = np.where(df_time_decay['timestamp_norm']==df_time_decay['max_timestamp'],
                                            0.6,
                                             np.where(df_time_decay['campaign_count'] == 2 ,
                                                      0.4,
                                                      np.random.uniform(0, 0.4, len(df_time_decay))))
```

	campaign	time_decay_prob	campaign_count	campaign_wt
0	73325	55.372536	173	0.320072
1	73328	149.377283	411	0.363448
2	83677	79.232022	224	0.353714
3	336258	11.452892	35	0.327225
4	442617	40.931504	115	0.355926
...	...	...	...	...
395	32385772	20.319692	58	0.350340
396	32398755	139.375407	457	0.304979
397	32398758	85.198512	296	0.287833
398	32405311	31.657545	96	0.329766
399	32452108	42.797180	128	0.334353

400 rows × 4 columns

Results:



Logistic Regression:

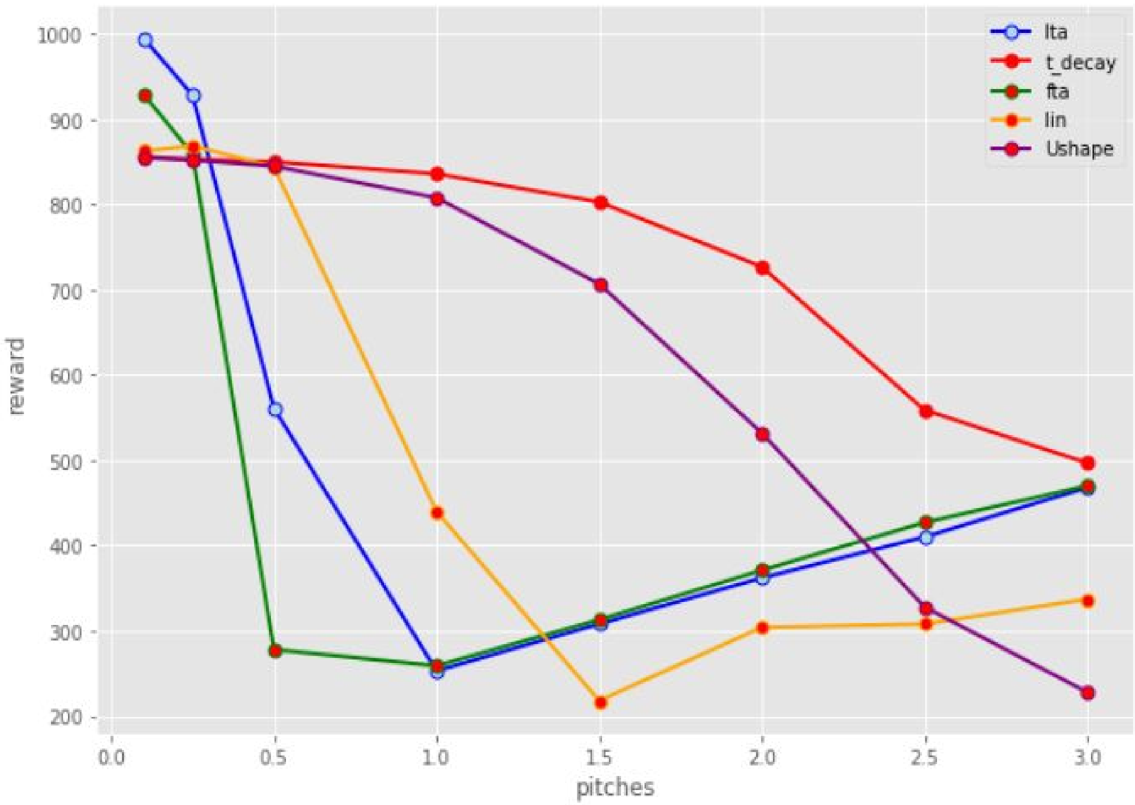
```
def features_for_logistic_regression(df):

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

    aggregation = {                    # aggregation specification for each feature
        '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
    )

x, y = features_for_logistic_regression(df6)
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size = 0.20)          # train-test split
x_train, x_val, y_train, y_val = train_test_split(x_train, y_train, test_size = 0.20)  # train-validation split
```

```
m = np.shape(x)[1]

model = Sequential()
model.add(Dense(1, input_dim = m, activation = 'sigmoid', name = 'contributions'))

model.compile(optimizer='rmsprop', loss='binary_crossentropy', metrics=['accuracy'])
model.fit(x_train, y_train, batch_size=128, epochs=10, validation_data=(x_val, y_val))
score = model.evaluate(x_test, y_test)
print('Test score:', score[0])
print('Test accuracy:', score[1])

#---------

Test score: 0.3732171668471283
Test accuracy: 0.8457783278327833
```

ROI Simulation:

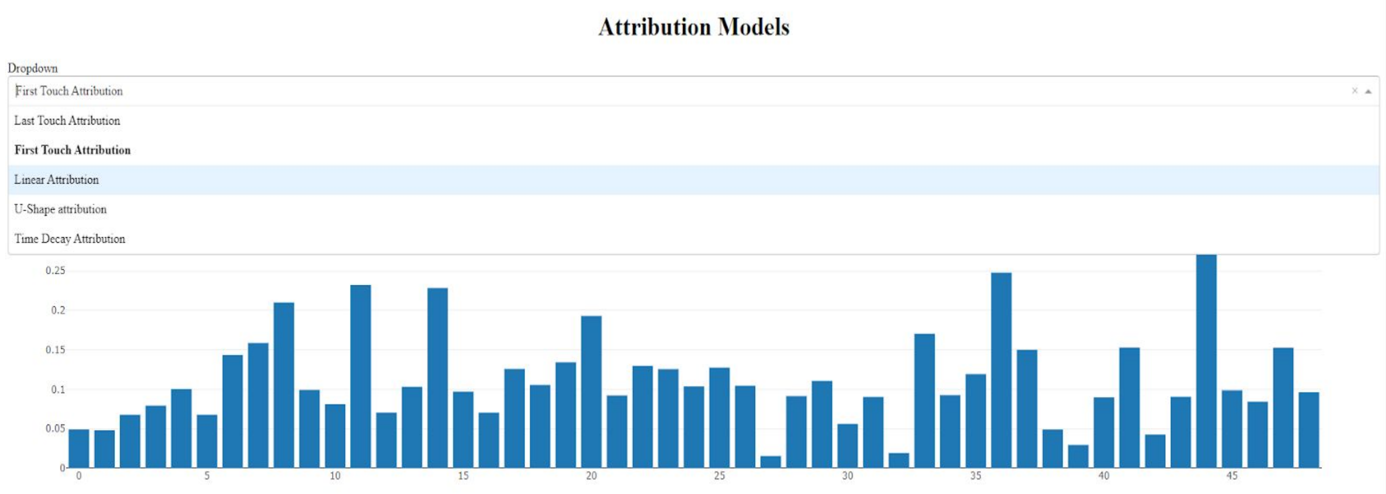
We have done simulations for 8 pitches to get the ROI for all the 5 models. Amongst the 5 models that we built Time decay and U shape models are giving the best ROI with respect to other 3 models.

```
<matplotlib.legend.Legend at 0x1d4c5318148>
```



Dash Visualization

For our team the tool we had got was a Dash. Using dash visualizations can be embedded inside a python web based app. Dash is a framework built on top of plotly and flask. For dash little bit of HTML is required to create layouts for the app. Using Dash we have created a drop down menu which could be used to select the attribution models.



We have also plotted a ROI graph using Dash.

Revenue Simulation
1000
800
600
400
200
0
0.5
1
1.5
2
2.5
3
lta
tdecay
fta
lin
ushape