

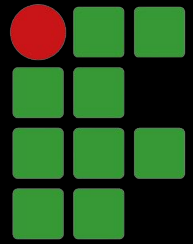
The background of the slide is a dark, semi-transparent overlay on a collection of colorful, 3D social media icons. These icons include Facebook 'f' logos, Twitter birds, and YouTube play buttons, scattered across the entire frame in various colors like blue, red, yellow, and green.

Predição de métricas de mídia digital

Pós Graduação - Ciência de Dados

IFSP - Campinas

Marcia Galeno, Rafael Barros, Vinícius Nascimento



Contexto

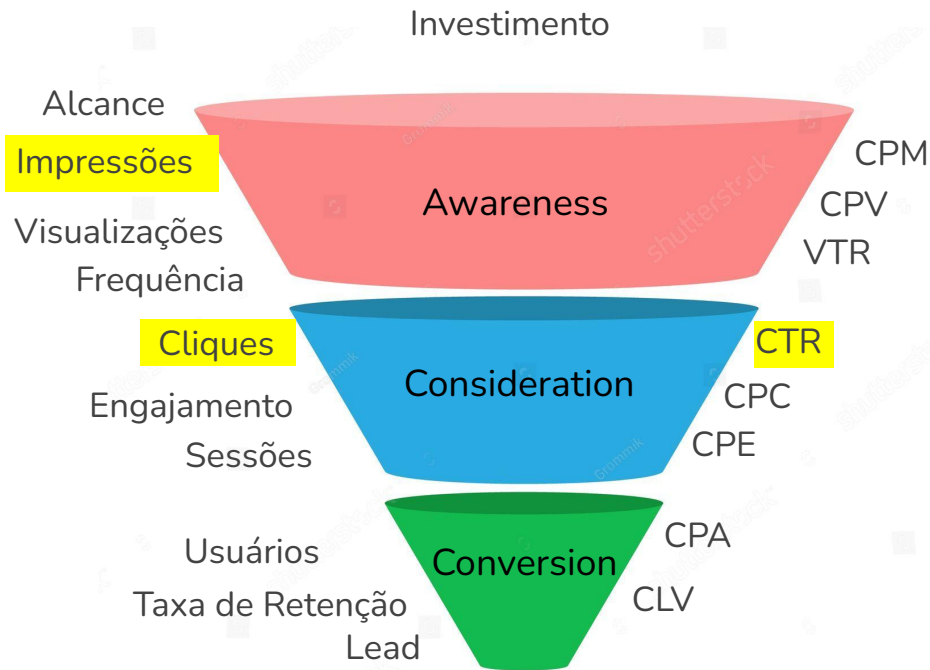
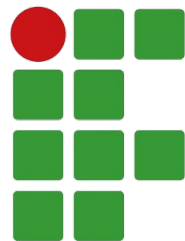
AWS

Análise exploratória

Aprendizado de Máquina



Contexto



$$\text{CTR} = \text{Cliques} / \text{Impressões}$$





Veículo: Meta

Funil: Conversão

Formato: Story Display

Sem Influencer

alcance
conversão



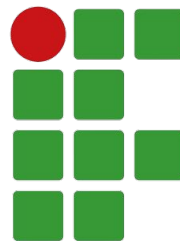
Veículo: Tiktok

Funil: Awareness

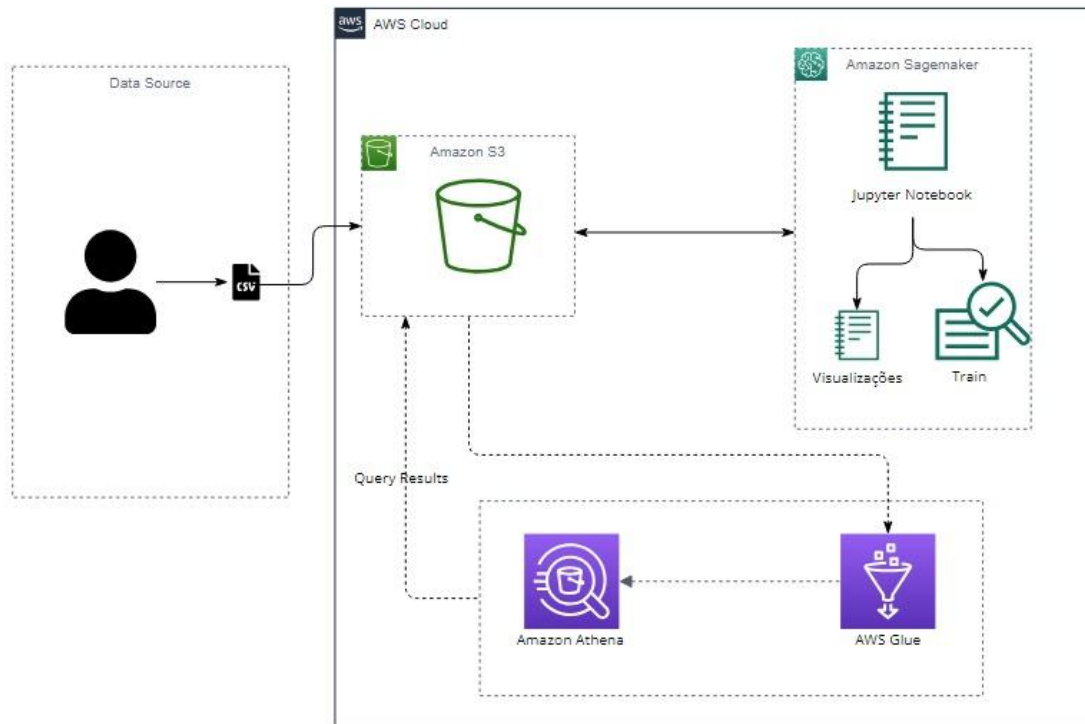
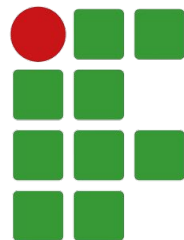
Formato: Vídeo 10s

Sem Influencer

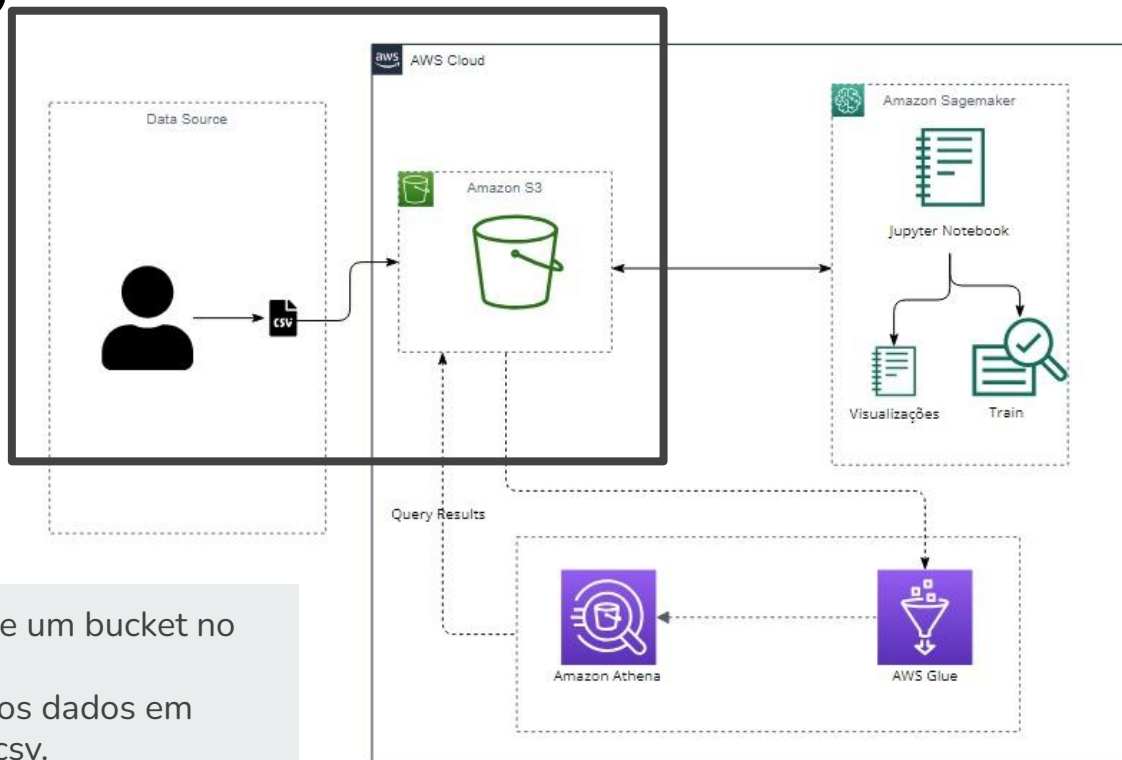
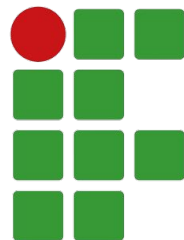
alcance
engajamento



AWS

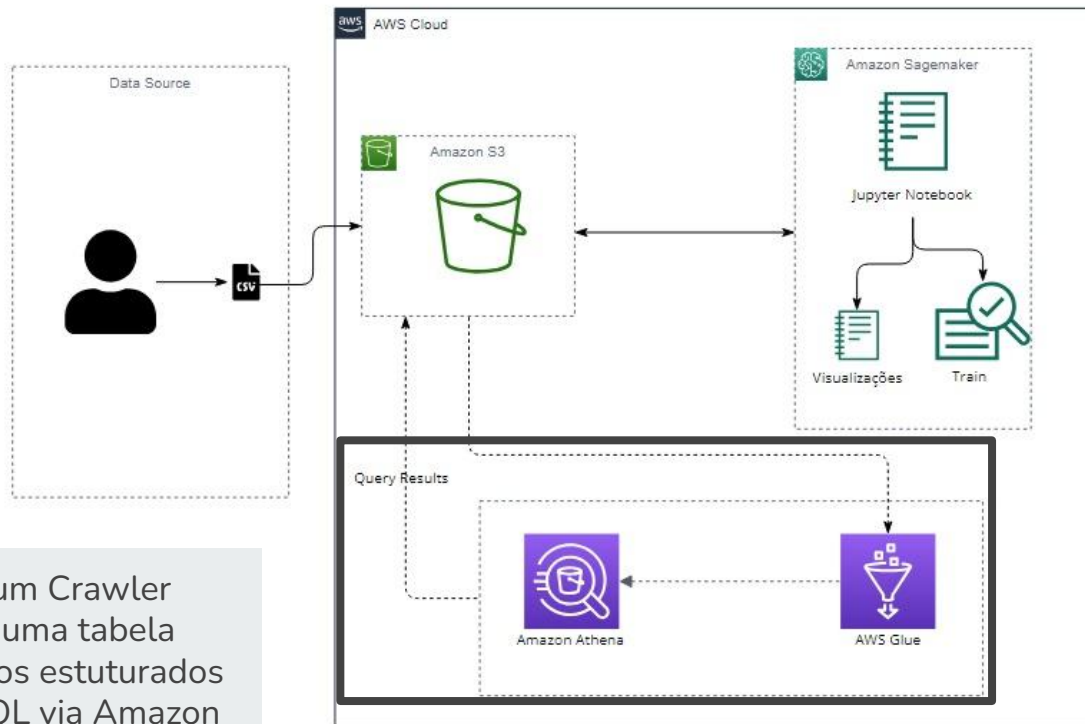
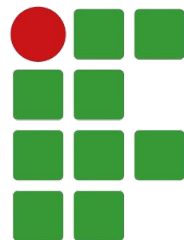


AWS



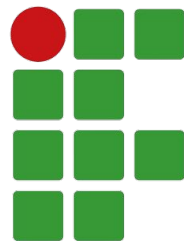
1. Criação de um bucket no S3.
2. Upload dos dados em formato .csv.

AWS



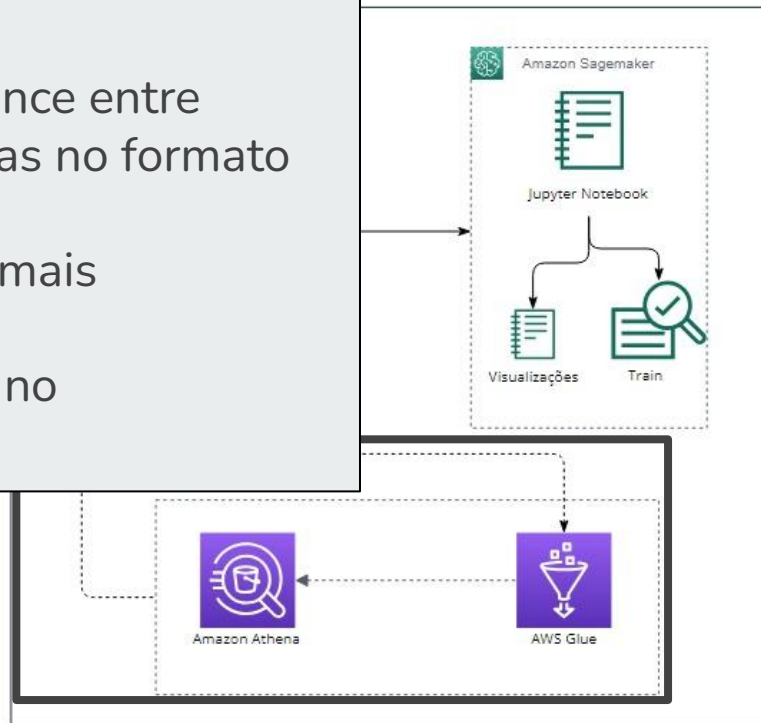
1. Criação de um Crawler
2. Geração de uma tabela com os dados estruturados
3. Consulta SQL via Amazon Athena

AWS

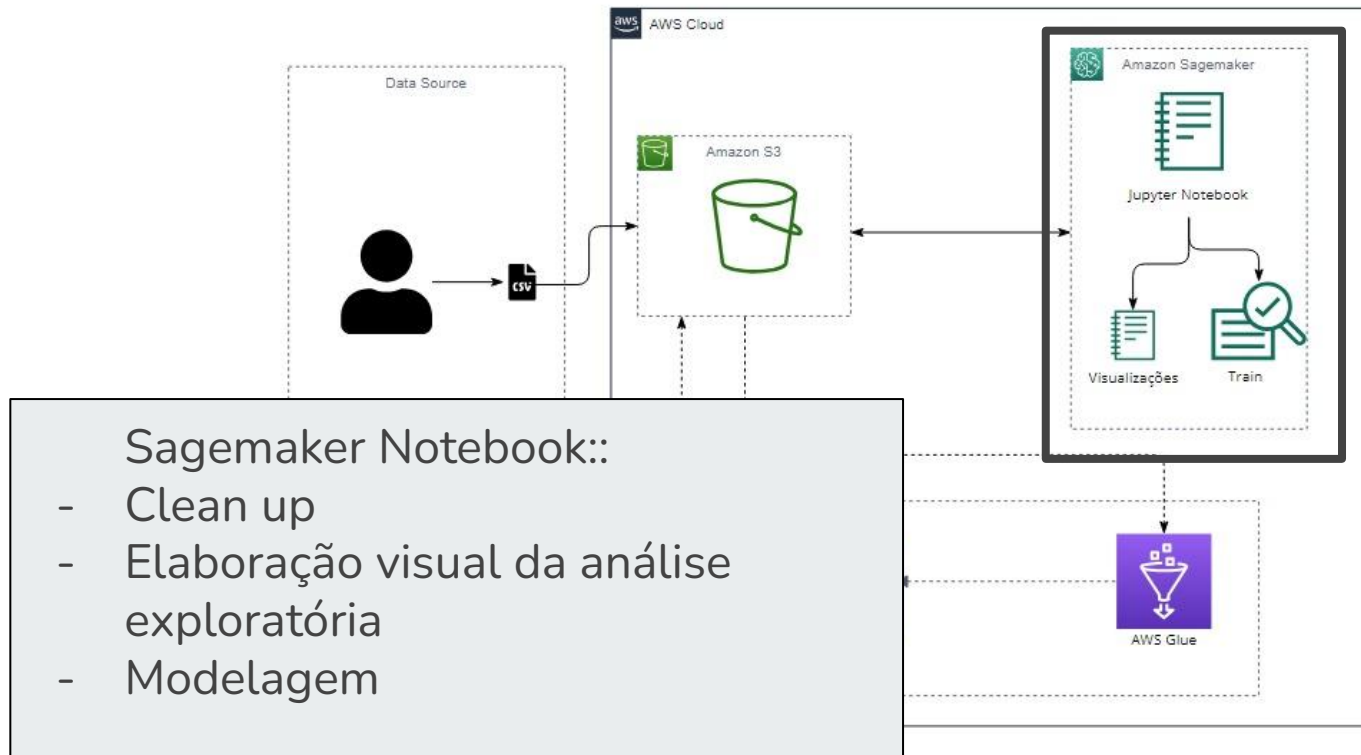
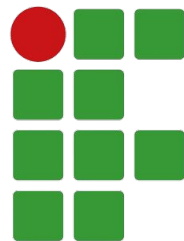


Queries:

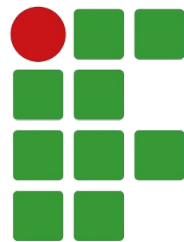
- Comparação da performance entre propagandas apresentadas no formato Video e Display.
- Os top vídeos de 3s com mais visualizações.
- O impacto de influencers no engajamento



AWS



Análise Exploratória



+R\$ 92M

Investimento

+30B

Impressões

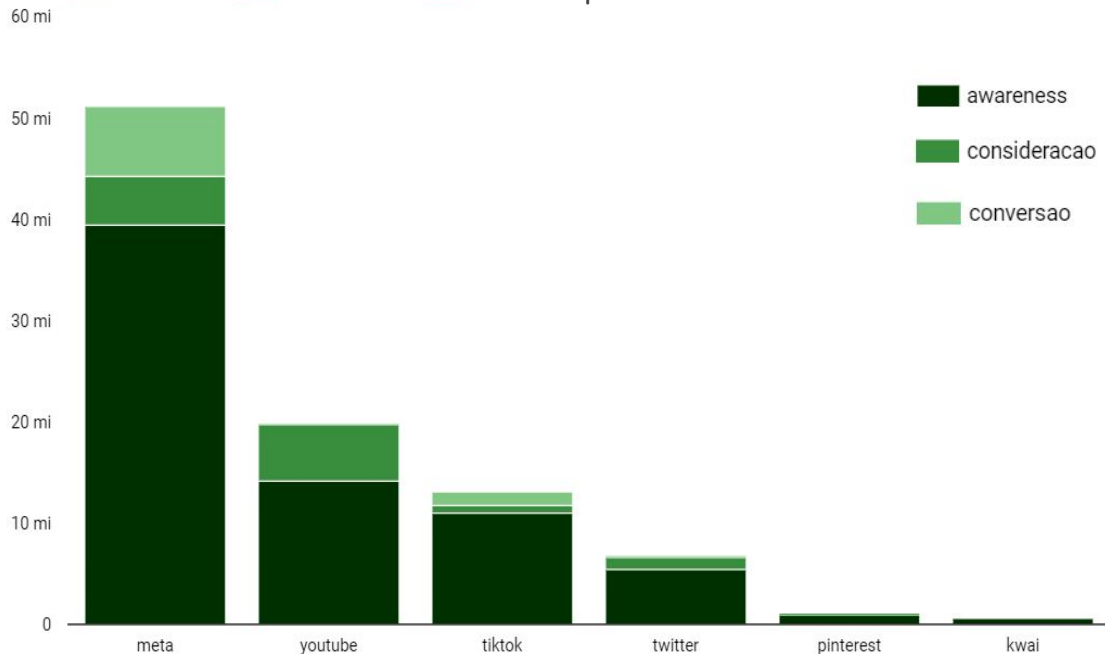
0,56%

CTR

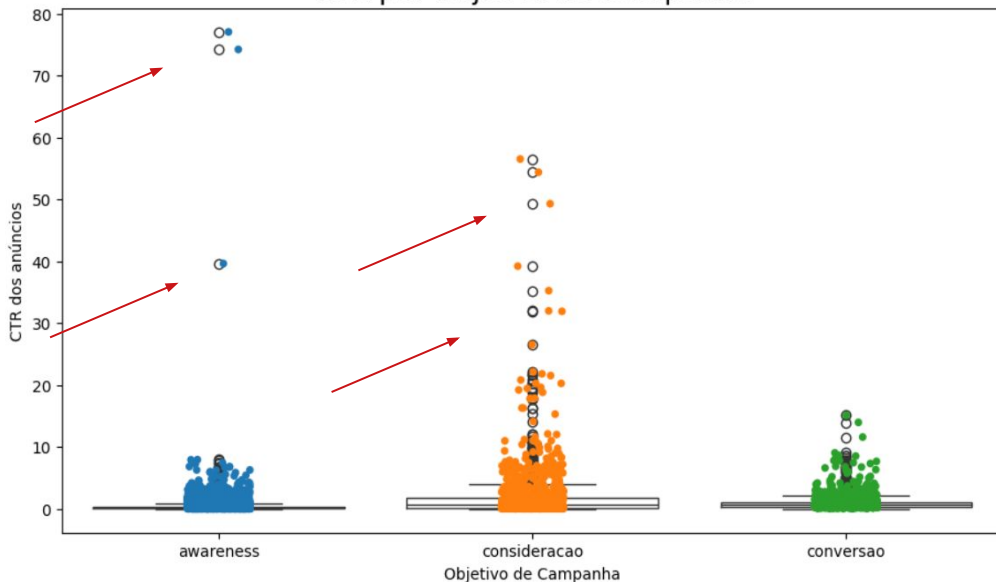
+172M

Cliques

Investimento realizado por Veículo e Funil



CTR por Objetivo de Campanha Com outliers

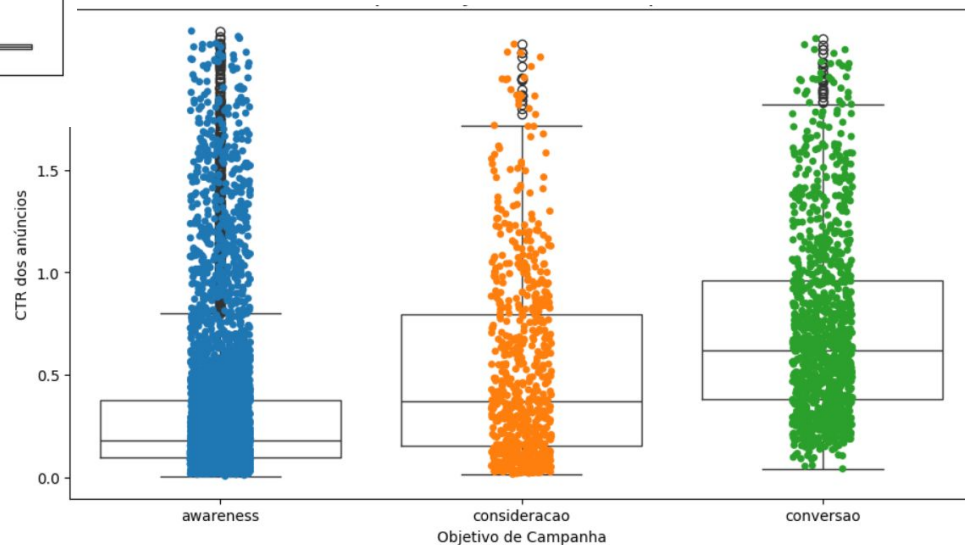


Consideração e Conversão têm CTR's maiores devido a segmentação do público.

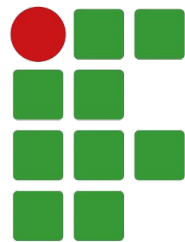
Muitos Outliers, que podem ter vindo de anúncios com engajamento orgânico ou baixo alcance.

Optamos pelo uso do Isolation Forest para identificar as anomalias que não se enquadram no padrão considerado normal da base.

Sem outliers



Análise Exploratória

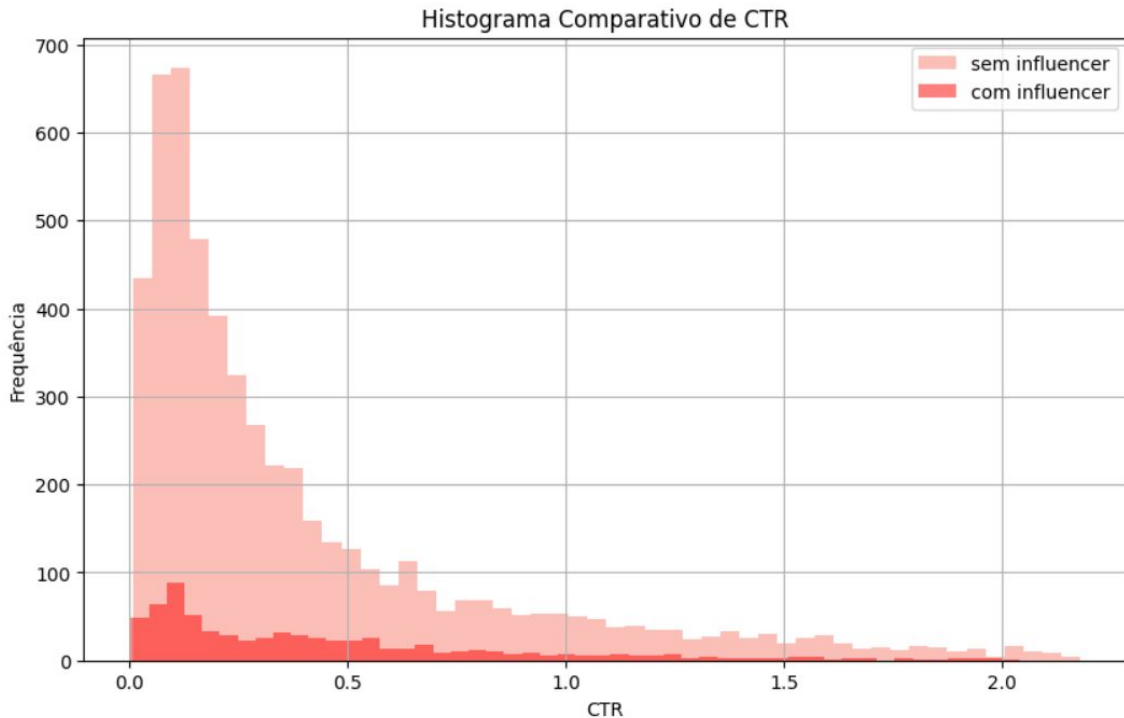


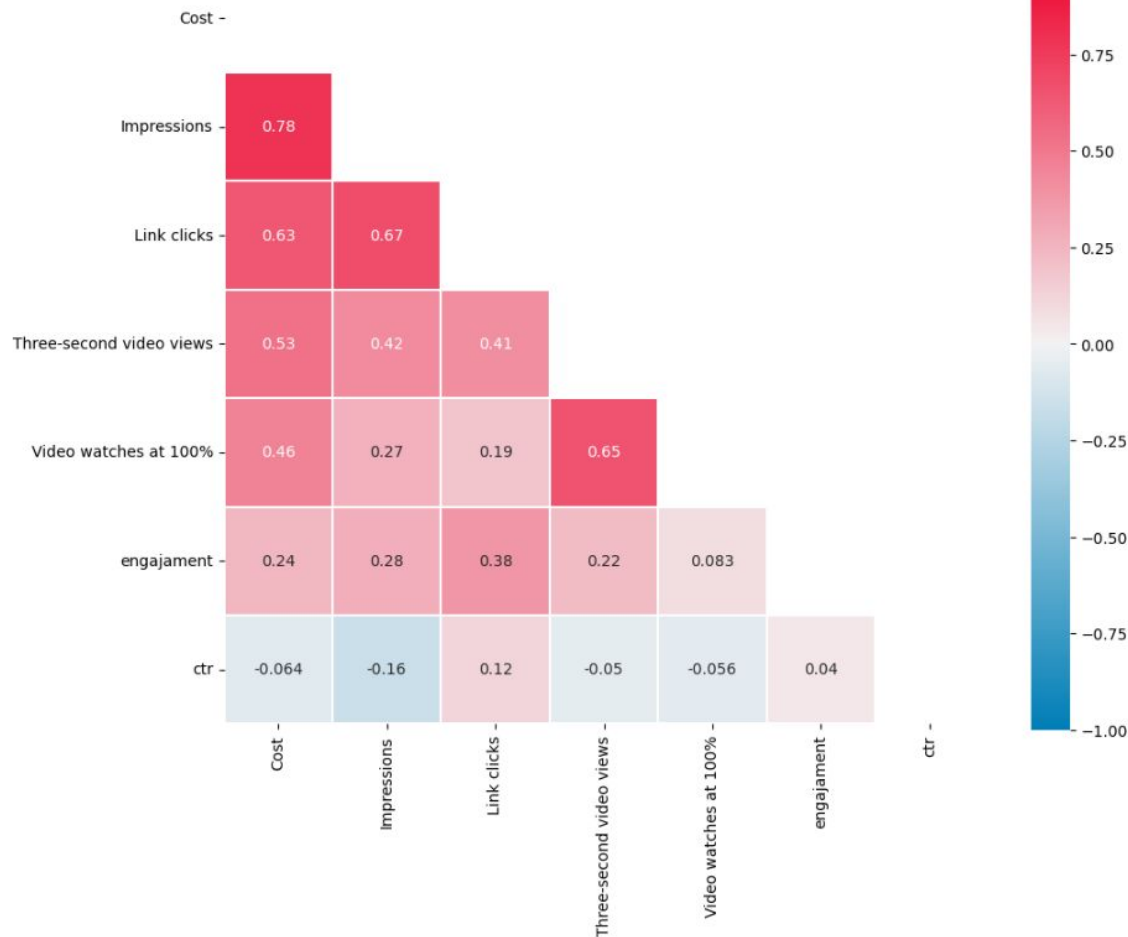
0,33%

CTR de anúncios **com**
influencers

0,25%

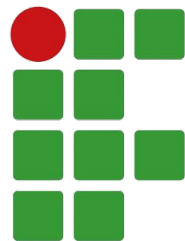
CTR de anúncios **sem**
influencers





Considerando que o comportamento dos anúncios pode variar de funil para funil, ou veículo para veículo e o Investimento não é responsável pela oscilação do CTR, seria mais preciso prever as Impressões e Cliques visto que possuem uma correlação mais forte.

Aprendizado de Máquina



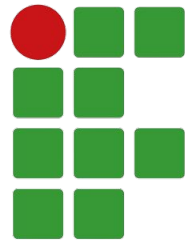
Após a codificação dos dados categóricos com o Dummies do Pandas, excluímos uma coluna de cada categoria a fim de evitar o Dummy Variable Trap.

```
df2 = df_no_outliers.iloc[:,[0,1,2,9,10,11,12]] #criando uma nova base apenas com as colunas que vamos usar para o modelo.  
#a partir daqui vamos codificar os dados categóricos da base usando o Dummies do Pandas.  
  
col = ['funnel', 'channel', 'format']  
df_encoded = pd.get_dummies(df2, columns=col) #criando novas colunas binárias a partir dos funis, canais e formatos.  
  
col_for_del = 'funnel_awareness', 'channel_meta', 'format_video 15s'  
for i in col_for_del:  
    del df_encoded[i]
```

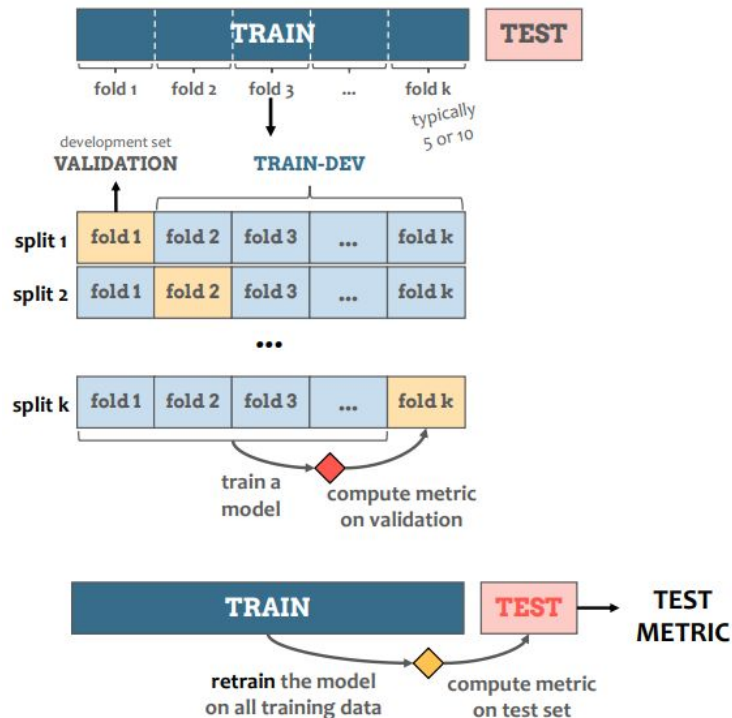
Dado o tamanho da base de cerca de 6 mil linhas, o conjunto de treinamento e teste foi dividido em 90/10.



Aprendizado de Máquina

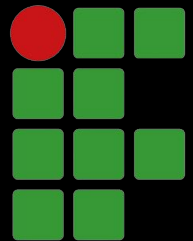


Devido a base possuir poucas instâncias, faz sentido utilizar um método de validação cruzada.



- Neste trabalho foi utilizado o KFold.
- 3 divisões (K).

Aprendizado de Máquina



Para os ajustes de hiperparâmetros optamos pela **Bayes Search**.

A otimização bayesiana é uma técnica baseada no teorema de Bayes, que descreve a probabilidade de ocorrência de um evento relacionado ao conhecimento atual. Quando isso é aplicado à otimização de hiperparâmetros, o algoritmo cria um modelo probabilístico a partir de um conjunto de hiperparâmetros que otimiza uma métrica específica. Ele usa análise de regressão para escolher iterativamente o melhor conjunto de hiperparâmetros.

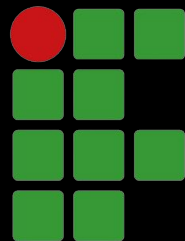
A métrica escolhida para avaliação de desempenho dos modelos foi R^2 .

Mesmo a normalização não sendo obrigatória em casos de regressão, optamos por simular o Random Forest na base com e sem o Robust Scaler. No Gradient Boosting não foi feita normalização.

[ref : https://medium.com/analytics-vidhya/comparison-of-hyperparameter-tuning-algorithms-grid-search-random-search-bayesian-optimization-5326aaef1bd1](https://medium.com/analytics-vidhya/comparison-of-hyperparameter-tuning-algorithms-grid-search-random-search-bayesian-optimization-5326aaef1bd1)



Aprendizado de Máquina



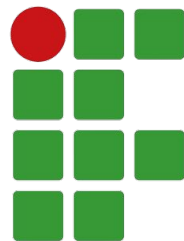
O modelo Random Forest processou dois target's simultaneamente quando atribuímos ao y uma lista.

```
bayes_rf_k.fit(X_ctr_treinamento, list(zip(y_clicks_treinamento, y_impressions_treinamento)))
```

O Gradient Boosting, por sua vez, exige o MultiOutputRegressor do Sklearn. No entanto, quando o unimos com o Bayes Search é necessário dividir os targets na busca pelos hiperparâmetros.



Random Forest Regressor



Definição do espaço de busca para os hiperparâmetros

```
n_estimators = [10, 20, 30, 35, 40, 45, 50, 80, 90, 100, 150]
max_features = ['sqrt', 'log2', None]
max_depth = [3, 5, 10, 13, 15]
min_samples_split = [2, 5, 10, 13, 15]
min_samples_leaf = [1, 2, 3, 4, 5]
bootstrap = [True, False]
```

Base sem Robust Scaler

Melhores Hiperparâmetros sem Kfold: `OrderedDict([('bootstrap', True), ('max_depth', 15), ('max_features', None), ('min_samples_leaf', 3), ('min_samples_split', 10), ('n_estimators', 30)])`
Melhores Hiperparâmetros com Kfold: `OrderedDict([('bootstrap', True), ('max_depth', 13), ('max_features', None), ('min_samples_leaf', 5), ('min_samples_split', 15), ('n_estimators', 100)])`

0.68

R² Clicks

0.66

R² Impressions

0.66

R² Clicks Kfold

0.64

R² Impressions Kfold

Base com Robust Scaler

Melhores Hiperparâmetros sem Kfold: `OrderedDict([('bootstrap', True), ('max_depth', 15), ('max_features', None), ('min_samples_leaf', 4), ('min_samples_split', 10), ('n_estimators', 90)])`
Melhores Hiperparâmetros com Kfold: `OrderedDict([('bootstrap', True), ('max_depth', 5), ('max_features', None), ('min_samples_leaf', 5), ('min_samples_split', 10), ('n_estimators', 20)])`

0.68

R² Clicks

0.67

R² Impressions

0.66

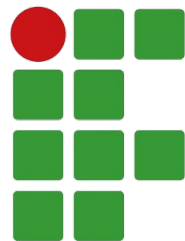
R² Clicks Kfold

0.69

R² Impressions Kfold



Gradient Boosting Regressor



Definição do espaço de busca para os hiperparâmetros

```
n_estimators = [ 50, 80, 90, 100, 150, 170, 180, 200]
learning_rate = [0.01 , 1.0, 'log-uniform']
max_features = ['sqrt' , 'log2', None]
max_depth = [3, 5, 10, 13, 15]
min_samples_split = [2, 5, 10 , 13, 15, 17]
min_samples_leaf = [1, 2, 3, 4 , 5, 7, 9]
```

Sem Kfold

Melhores Hiperparâmetros sem Kfold: OrderedDict([('learning_rate', 0.14170409680320734), ('max_depth', 13), ('max_features', 'sqrt'), ('min_samples_leaf', 7), ('min_samples_split', 2), ('n_estimators', 50)])

target Impressions sem Kfold: OrderedDict([('learning_rate', 0.13216840404382582), ('max_depth', 3), ('max_features', 'sqrt'), ('min_samples_leaf', 9), ('min_samples_split', 17), ('n_estimators', 200)])

0.75

R² Clicks

0.72

R² Impressions

Com Kfold

target Clicks: OrderedDict([('learning_rate', 0.2227573224949704), ('max_depth', 5), ('max_features', 'sqrt'), ('min_samples_leaf', 9), ('min_samples_split', 2), ('n_estimators', 50)])

target Impressions: OrderedDict([('learning_rate', 0.07852256558632155), ('max_depth', 15), ('max_features', 'log2'), ('min_samples_leaf', 7), ('min_samples_split', 13), ('n_estimators', 150)])

0.75

R² Clicks Kfold

0.73

R² Impressions Kfold

Obrigada!
Obrigado!

