Lead scorer

2022 february

KSchool Master Data Science

Igor López Reigosa

1. Intro: What, why & previous works 3
What's this project about? 3
Previous works, state of the art 3
Why this lead scorer? 4
2. Data description & dictionary 5
Data sources 6
a) Corporate data 6
b) GA data 6
c) SQL query on Redshift 7
3. Files structure 8
4. Methodology and techniques 9
Statistical techniques 9
Machine learning techniques 9
4.1. Data acquisition 10
4.2. Data cleansing & preparation 10
4.3. Model testing and optimizing 13
4.4. Visualization 17
5. Conclusions: 19
6. How to run the project 20
6.1. Instructions for running the project: 20
6.2. Front end user instructions 20
7. Bibliography: 21

1. Intro: What, why & previous works

What's this project about?

This lead scorer is a machine learning project that tries to predict the chances of a generated lead to become a booking (commercial visit appointed to the customer's house). Predictions consists of a probability that will be used as a score for different usages:

a) Reingestion in Google & Facebook ad servers to let the algorithms learn.
b) Add a score when passing leads to contact centre that helps them to prioritize.
c) Develop a simple / replicable methodology to be scaled in other countries where the company operates.

As a context, the company is a worldwide advertiser, market leader on its segment, present in 16 countries on Europe and Latam, and seeking for expansion on new markets though different brands. Its lead generation operation consists of generating prospects through any digital channel, mainly website and company assets. Once generated, the lead goes to Contact Centre, where it enters in different calling lists. When contacted, the goal is to get the user to appoint a commercial visit with the company agents, which will proceed to make a proper assessment and therefore offer to get the alarm installed.

Previous works, state of the art

Having said that, the 'new thing' this model tries to bring to is the fact of having been developed on a lack of personal and historical user data. Traditional scorer models have been developed from user data, both outside company assets (social media and ads-based mainly) and inside company assets (browsing history, ecommerce purchases, clv), but there is no market expertise for models with such a few of user data.

Why this lead scorer?

The choice of this topic mainly attends to three big reasons:

1. First, to try **to gain efficiency on the company lead generation teams**, by contributing with machine learning data to the two key cores of the operation:
    a. Ad serving: audiences and auctions occur on a lead basis, with the platforms 'blind' to further data.
    b. Contact centre: Historically this teams organize calls based on the channel the leads come from, but not seeing other indicators such as quality.

2. Also, I wanted to **make the most of the technology** that we have been developing and launching at a corporate level. In fact, the company end-to-end DWH built during 2021 has been used for leads and bookings data. When ideally the model goes live, it would also use company technology, as it would use our existing processes to integrate the predictions on our ad servers via offline events APIs.

3. And finally, it was key to test the capability of modelling in a scenario of non-personal, cookie-based data. Two big reasons behind this:

    a. Web and **digital data ecosystem have been notably shrinked** by regulations protecting personal data and restricting tracking cookies. This is perceived as a threat by our local teams, so new solutions need to be explored.

    b. By its nature, the **company digital lead generation operation is scarce on personal data** (and perhaps qualified in general?), as there is no login or identification action. Historical user data is neither available, as the business model is not recurrency. The only personal data needed is a phone numbers, but legal commitments make impossible to work with this.

2. Data description & dictionary

Feature	Type	Source	Details
event_to_date	object	company e2e	lead date
lead_id	object	company e2e	lead id
clientid	float64	company e2e	Google Analytics client_id
entries_count	int64	company e2e	web counter – entrances
starts_count	int64	company e2e	web counter – start quote
postcode_count	int64	company e2e	web counter – reach postcode
phone_count	int64	company e2e	web counter – reach phone
lead_count	int64	company e2e	web counter – reach postcode
device_browser	object	company e2e	browser, from user agent
device_browser_size	object	company e2e	browser size, from user agent
device_browser_version	object	company e2e	browser version, from user agent
device_category	object	company e2e	mobile / desktop, from user agent
device_marketing_name	object	company e2e	device model, from user agent
device_mobile_model	object	company e2e	device model, from user agent
device_operating_system	object	company e2e	device OS, from user agent
device_operating_system_version	object	company e2e	device OS version, from user agent
device_mobile_branding	object	company e2e	device brand, from user agente
source	object	company e2e	source parameter from url
medium	object	company e2e	medium parameter from url
campaign	object	company e2e	campaign parameter from url
campaign_cd	object	company e2e	campaign parameter from web data layer
is_booking	int64	company e2e	target feature, binary
step0	object	Google Analytics	quoter step 0 answer
step1	object	Google Analytics	quoter step 1 answer
step2	object	Google Analytics	quoter step 2 answer
step3	object	Google Analytics	quoter step 3 answer
step4	object	Google Analytics	quoter step 4 answer
step5	object	Google Analytics	quoter step 5 answer
step6	object	Google Analytics	quoter step 6 answer
step7	object	Google Analytics	quoter step 7 answer
step8	object	Google Analytics	quoter step 8 answer
step9	object	Google Analytics	quoter step 9 answer
step10	object	Google Analytics	quoter step 10 answer

Data sources

a) Corporate data

Due to the amount of available data (both because since when it as implemented and the volume of leads & bookings), **this pilot has been built for Argentina with 2021 data** (July to December).

Data **from the company end-to-end** comes mainly from the user agent. It also grabs the client_id from Google Analytics (anonymous identifier on GA platform with a 30 days expiration), which is needed for joining GA data. All this data is only captured with the appropriate user consent on tracking. The target feature 'is_booking' also comes from this data source. It describes if a contacted lead has appointed a commercial visit to the users home in order to get the alarm installed (1 for positive, 0 for negatives).

This company end-to-end DWH has been built as a set of multiple tables for the different business areas involved, but also for different data purposes as it's the same structure for all the countries. As it can't be accessed from outside the company, the query used is pasted next as a reference, and the needed files to run this repo are provided outside.

b) GA data

Google Analytics collects all the consented interactions with the site quoter, which is 'the producer' of leads (all the converters fill the quoter in around a 90%). The quoter is different for business and home alarms, but data collection occurs the same way in any scenario (how answers are tracked by GA on events remains the same).

I've used Google Analytics API on its v4, for Python. There are two scripts that extract:

1) All the leads tracked with its client_id and lead_id.
2) All the event categories (type of funnel), actions (questions) and labels (answers) for all the converters (this was isolated by extracting only events for this user segment) by client_id. As this query returns almost a million of rows and due to how this API works, it was needed to work with paginated results.

These scripts can be seen on the 'Get_Data' notebook. They are merged into a single file on 'Clean_Data', to be later analysed.

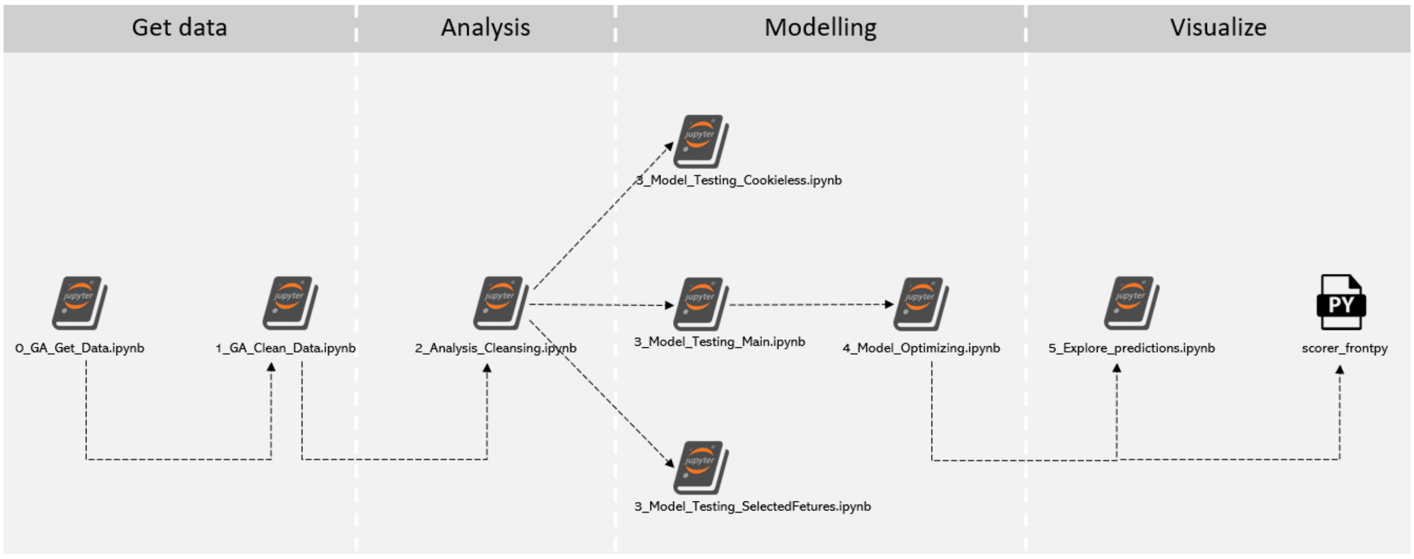
As Google & company credentials are needed in order to use corporate Google Analytics (see references to 'client_secrets.json' on the notebooks, these files are also provided from outside the repo.

c) SQL query on Redshift

```
select
        *
From
        (
        select
                to_date(um.event_date, 'YYYY-MM-DD HH24:MI:SS', true) as lead_date,
                um.lead_id,
                um.clientid,
                count(CASE WHEN um.event_name = 'uni__entry' THEN 1 END) as entries_count,
                count(CASE WHEN um.event_name = 'uni__funnel_start' THEN 1 END) as starts_count,
                count(CASE WHEN um.event_name = 'uni__postcode' THEN 1 END) as postcode_count,
                count(CASE WHEN um.event_name = 'uni__phone_number' THEN 1 END) as phone_count,
                count(CASE WHEN um.event_name = 'uni__lead' THEN 1 END) as lead_count,
                um.device_browser,
                um.device_browser_size,
                um.device_browser_version,
                um.device_category,
                um.device_marketing_name,
                um.device_mobile_model,
                um.device_operating_system,
                um.device_operating_system_version,
                um.device_mobile_branding,
                um."source",
                um.medium,
                um.campaign,
                um.campaign_cd,
                case
                        when ms."booking/sale_creation_datetime" is not null then '1'
                        else '0'
                end as is_booking,
                to_date(ms."booking/sale_creation_datetime", 'YYYY-MM-DD HH24:MI:SS', true) as booking_date,
        from
                mkt_verisure.unified_model_ar_events as um
        left join mkt_verisure.master_ar as ms on
                ms.dialer_lead_id = um.lead_id
        group by
                um.clientid,
                um.lead_id,
                lead_date,
                um.device_browser,
                um.device_browser_size,
                um.device_browser_version,
                um.device_category,
                um.device_marketing_name,
                um.device_mobile_model,
                um.device_operating_system,
                um.device_operating_system_version,
                um.device_mobile_branding,
                um."source",
                um.medium,
                um.campaign,
                um.campaign_cd,
                ms."booking/sale_creation_datetime")
where
        (lead_date >= '2021-07-07' AND lead_date < '2021-12-31')
        and lead_count >= 1
order by
        lead_date asc, entries_count desc
```

3. Files structure

This is how the files and notebooks are organized:


Get data
Analysis
Modelling
Visualize
0_GA_Get_Data.ipynb
1_GA_Clean_Data.ipynb
2_Analysis_Cleansing.ipynb
3_Model_Testing_Cookieless.ipynb
3_Model_Testing_Main.ipynb
3_Model_Testing_SelectedFetures.ipynb
4_Model_Optimizing.ipynb
5_Explore_predictions.ipynb
PY
scorer_frontpy

Some caveats:

- Notebooks 0 to 3 produce outputs (csv files) that are used as an input on each notebook's next file.
- Notebook 3 is 'duplicated' on cookieless (no features from GA) and non-cookieless all features. Cookieless was a dead end that was abandoned.
- Both 'Explore_predictions" and "FrontEnd" are built from predictions csvs.
- "FrontEnd" uses another Python file where the functions used are stored.

4. Methodology and techniques

Statistical techniques

- **EDA**: Exploratory Data Analysis to identify anomalies, formulate hypothesis about features and understand how them are related and well informed.
- **Basic feature engineering**: Some of the raw data features have to be transformed in order to get valuable inputs from them. This includes some feature reduction.
- **Feature selection** to have a dataset reduced in columns, so model testing can be run faster and noise is decreased from the main dataset.

Machine learning techniques

- **Pipelines and GridSearchCV** to compare models performance and explore variability when hyperparameter tuning.
- **Classification**: Different classification models such as Decission Trees, KNN, or XGBoost have been trained and tested in order to understand pros and cons on each of them.
- **Deep learning**: A neural network is built and trained to have a comparison against how classification models perform.

Methodological approach

Given the nature of this project (succinct on features, all categorical, imbalanced, need to be highly explainable and scalable), a few basic principles have been followed:

- **Explore all options** on feature selection and encoding.
  - Model testing is performed both for main transformed data (all features, all rows), cookieless (non GA features) and also with reduced data.
  - It also is built for test models both with features one hot encoded, regular target encoded and cv target encoded.
- **The easier, the better**: when different solutions end up showing very similar results, we try to select always the simplest one in terms of computing and understanding.

4.1. Data acquisition

As described before, **corporate data comes from company DWH**, being extracted through a SQL query on Redshift. This is only a matter of querying and uploading csv files.

Regarding **GA data**, some additional tasks had to be done:

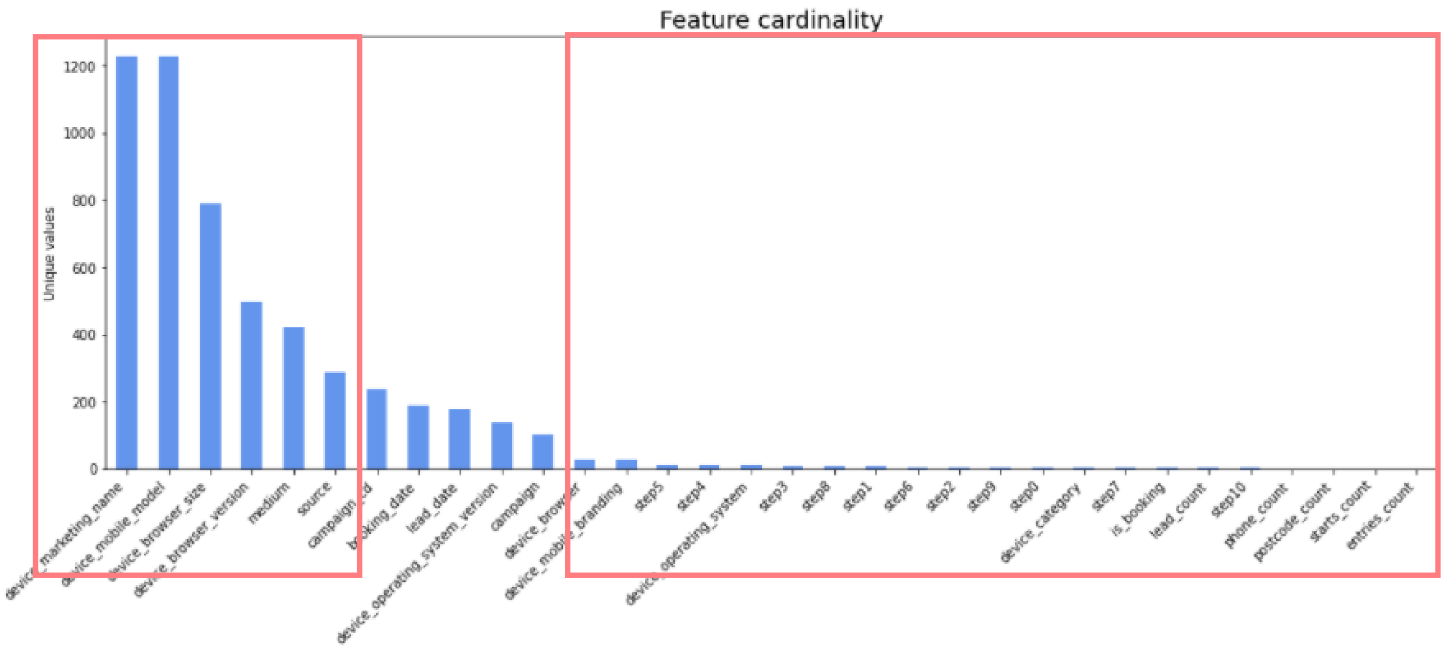
1. Understand data model on quoter questions and answers.
2. Create a segment to isolate data from lead converters. This is something that must be done on the platform UI and later used on the script through its segment id.
3. Loop over paginated query results and extract all the rows. The Q&As query has almost a million of rows, which the API only able to return in chunks of 10K.
4. Create data frames from the results and merge them into a single GA data object.

After corporate and GA data acquisition and before starting feature analysis, both are merged into a single one. This can be easily done merging them on the lead_id field.

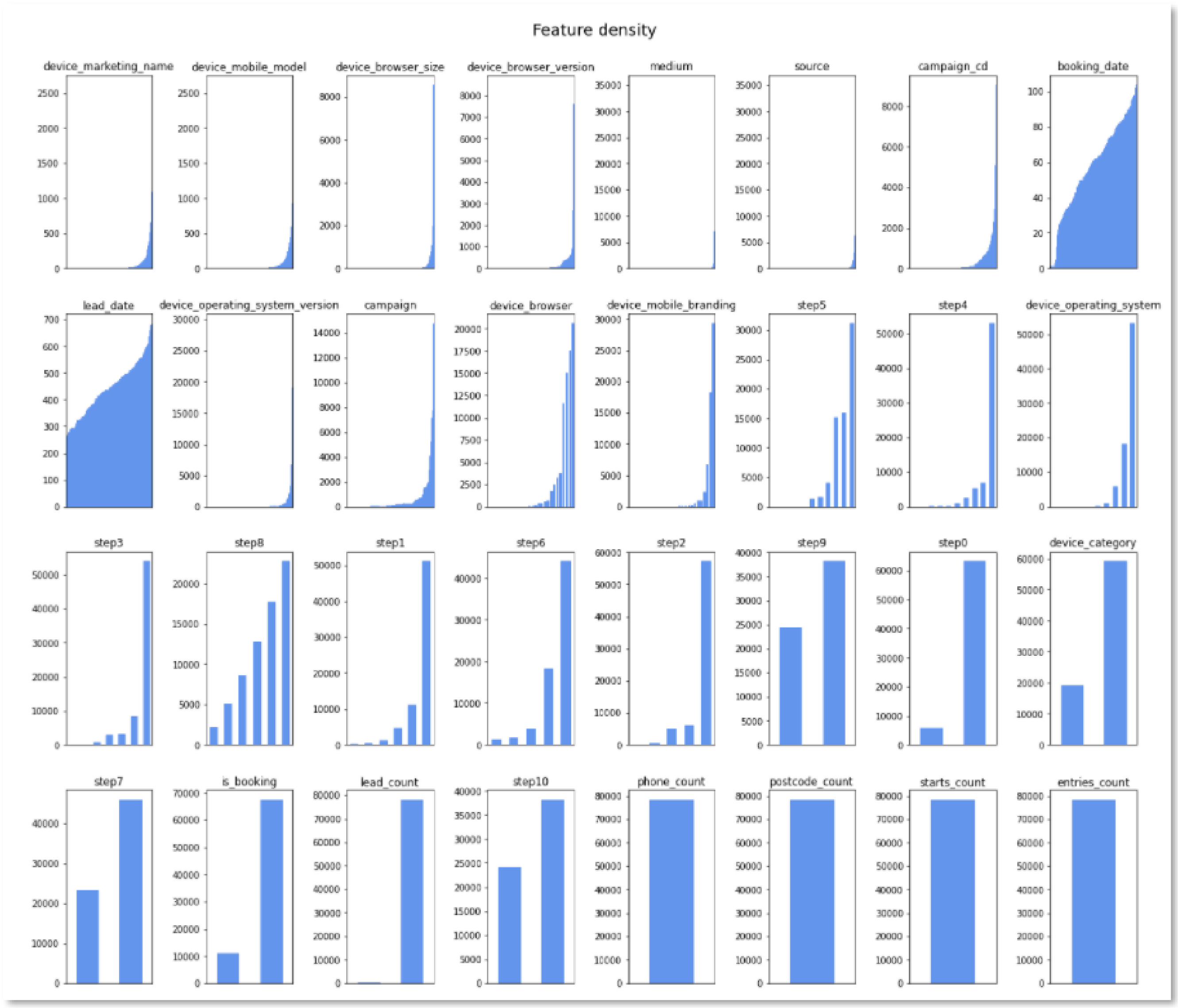
4.2. Data cleansing & preparation

Here the objectives are a) understand features relations and importance to b) produce a preprocessing that outputs a data frame ready to be modelled. EDA quickly reveals some fundamental insights:

- Data features are basically categorical.
- Data is imbalanced. The target (0, 1) is positive in almost 14% of the cases.
- Features can be grouped on a) high cardinality and b) low cardinality ones.



This is how features were formed in terms of cardinality and density:

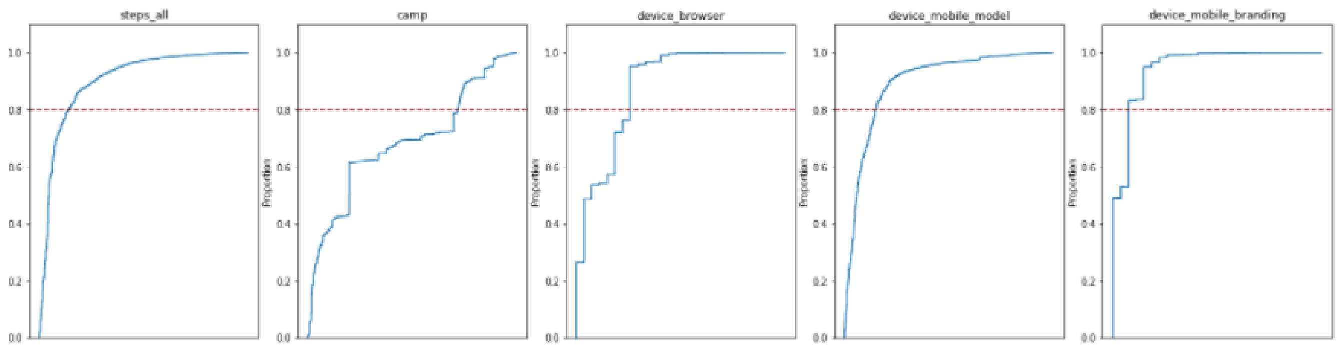


In order to understand features relations and importance and get a final data frame ready to be modelled, actions done:

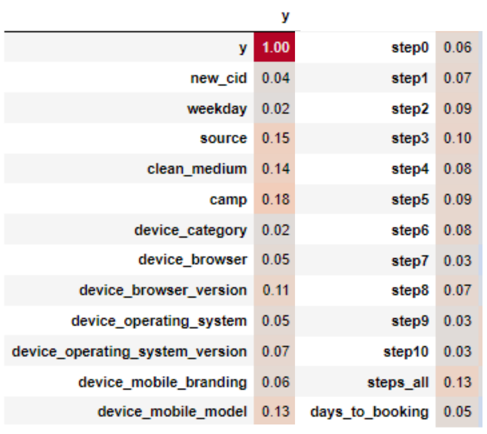
- **Transform non-informative, bad-formed features** into conceptually logical ones.

Base feature	New feature	Description
clientid	new_cid	Boolean, same user or not
lead_date	weekday	Day of the week
lead_date & booking_date	days_to_booking	AVG campaign days until booking, if exists
all counters	web_events	Boolean, multiple quoter interactions or not.
all steps	steps_all	String concatenating all quoters answers

- **'Reduce' cardinality on specific features** by 'cutting' on the values that concentrates until the 80% of all the occurrences. Last 20%, longtail segregated into multiple, few repeated values, are grouped by a dummy 'reduced' value. This is mainly intended to not overload one-hot-encoder with unnecessary potential columns.
  This approach was **abandoned once we see that it reduces the correlation between features in around 15%**. In addition, we'll have the dataset of only selected features to speed up ohe and model testing.



- **Explore correlation by 'dummy' encoding features**. A correlation matrix (one for normal features and other for the just reduced ones) is built previous Target Encoding all columns, as all of them were categorical, object type. Target Encoder hyperparameters can be surely improved, but at first sight seems to be really low correlation between variables. None of he features shows more than a 20% correlation with the target, so it seems really complex that any model will be able to accurately give predictions.



	y
y	1.00
new_cid	0.04
weekday	0.02
source	0.15
clean_medium	0.14
camp	0.18
device_category	0.02
device_browser	0.05
device_browser_version	0.11
device_operating_system	0.05
device_operating_system_version	0.07
device_mobile_branding	0.06
device_mobile_model	0.13
step0	0.06
step1	0.07
step2	0.09
step3	0.10
step4	0.08
step5	0.09
step6	0.08
step7	0.03
step8	0.07
step9	0.03
step10	0.03
steps_all	0.13
days_to_booking	0.05

- **Explore feature importance**. As many features are useless, but also to have a smaller dataset to play with, we did some **feature selection via RFECV and chi2 using KBest**. We had different outputs (features), that had been compared and a final list of features was elaborated from there.

```

    elif feature_selection:
        df_clean = df_clean.filter(items=['camp', 'clean_medium', 'device_browser_version',
                                          'device_mobile_model', 'steps_all', 'is_booking'])

```

It's also revealing how RFECV shows 1 feature as the optimal number of features to use. Although it can be biased by a 'dummy' previous target encoding and model selection, but it's a good proxy to understand how features are adding more noise than predictive capabilities.

4.3. Model testing and optimizing

Different models are tested and compared on block 3 'Model Testing'. After that, a single model is trained and optimized on block 4 'Model Optimizing'.

We know from block 2 'Analysis' that our features have small predictive power, so how we treat them becomes especially important. 'Model Testing' aims to explore:

1. Differences on **modelling with different datasets** (normal, selected features, cookieless).
2. Differences on **modelling with different encoding techniques** (ohe, target encoding, target encoding with cross validation).
3. Understanding **models pros and cons** for this classification purpose.

Tested models (other have been abandoned) using sci-kit learn and tensorflow:

1. Decision Tree Classifier
2. Random Forest Classifier
3. KNeighbors Classifier
4. XGBoost
5. Boosting DTC
6. Voting Classifier from top performers
7. Neural network

Notebooks from this block (named 3_) follow this structure:

1. **3 different encodings/pre-processing** are performed at the beginning (OHE, TE, TECV) to do them a single time and therefore avoid unnecessary encoders computing when testing models.
2. Evaluation **metrics are based on performance over test set**, although also accuracy and F1 figures on train set are reported to get an idea of overfitting.
3. Each **model is approached first with a** base, naked model fit & predict (base) for each encoder, **but also with a small GridSearchCV** on the main hyperparameters (tuned) to explore how it behaves. Target Encoder CV uses the GS best params from the regular Target Encoder.
4. At the end, a **comparison between the different results** (models and the encoding they use) is built.

Each model has:

- Base model on OHE
- Base model on TE
- Base model on TECV
- Tuned model on OHE, previous GridSearchCV
- Tuned model on TE, previous GridSearchCV
- Tuned model on TECV using TE GridSearchCV best params

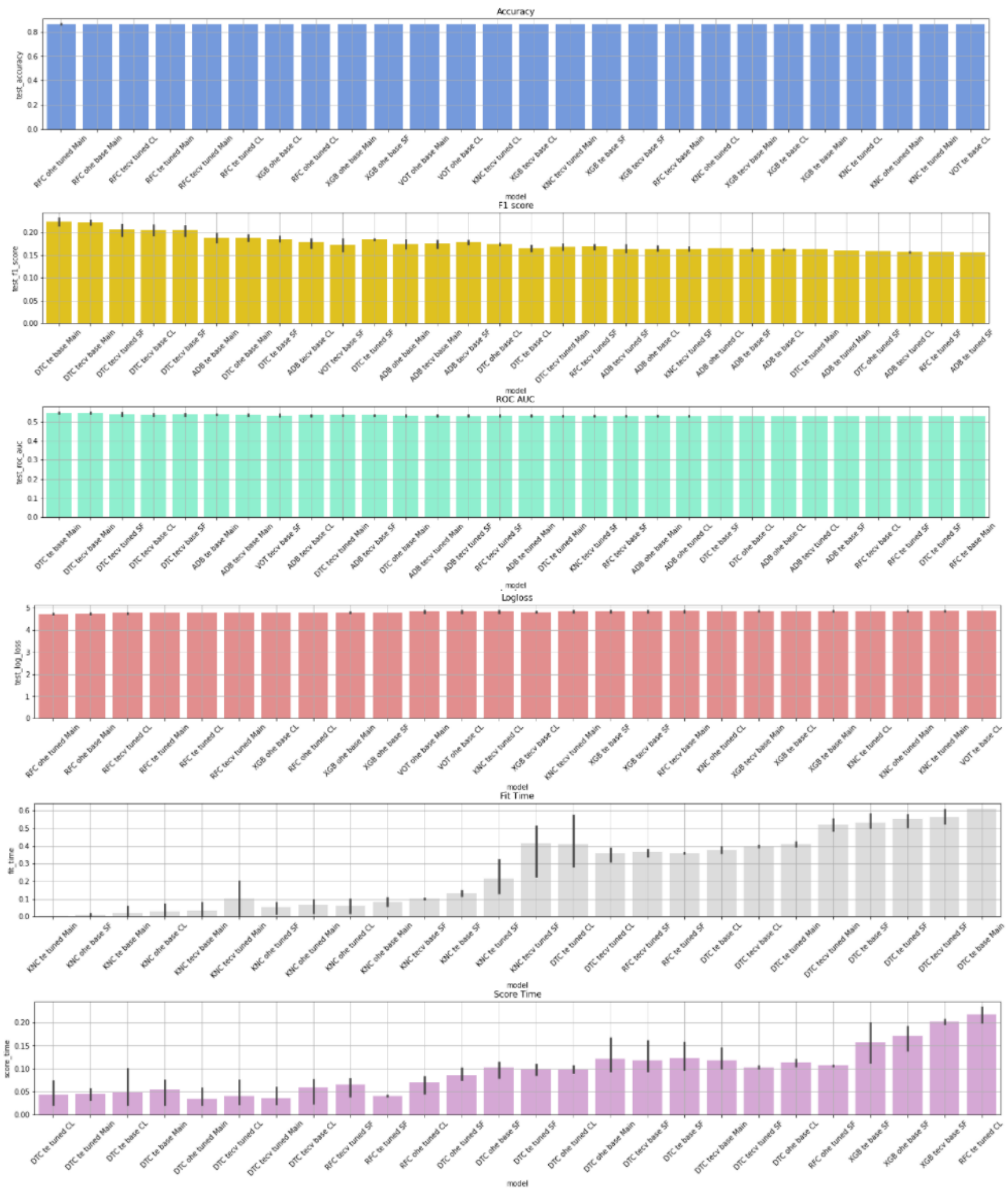
The effort of tuning all models performing GridSearchCV for their different approaches may seem excessive, but it has been used as the way to learn and explore them. It has been key for some model's performance, as for example XGBoost, which was one of the worse models on base executions, but one of the best when tuned. In fact, GridSearchCVs that can be seen on the notebooks are the last results after several iterations searching what's important.

The neural network did not get better performance than the other models.

Uploaded notebooks to the repo are a summary of all the models tested, as **other models have been abandoned** to gain speed & replicability. Additional insights:

- **Linear models logically don't work well** for this data, both normal and selected features ones. Some of them are also expensive, such as LSVC.
- **Naive Bayes classifier was explored via GaussianNB**, but with poor results, probably because features are not independent from each other.
- **Gradient Boosting Classifier easily makes the kernel die** when tuning the model. So, it was abandoned on final testing.

On block 4 'Model Optimizing' we compare the performance of all the tested ones to select a final model:



Attending to above results, chosen model was DecisionTreeClassifier on the whole data set ('Main' one, 'data_files/preproc_clean.csv'):

- All models show accuracy close to 80%
- DTC models show the best F1 scores.
- They are also light and fast, so it allows to more tuning with less computing effort.
- Biggest dataset was chosen as it's performing well but also because it allows to perform feature selection on the final pipeline.

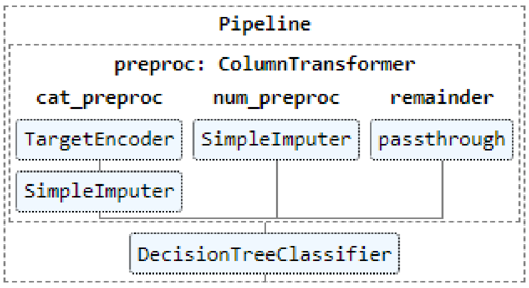
From the model selection to the end, block 4 'Model Optimizing' is intended to get the most of the chosen model. Unlike the previous block we put together **pre-processing and modelling tasks on the same pipeline.**

Each iteration includes pre-processing and adds different tasks to be tested until the final model definition. At the end, we save the final model and its predictions both on train and test sets to be later used for visualization.

Different iterations can be summarized as follows:

- **1st iteration explores a wide params** net, from target encoder to classifier options.
- **2nd iteration explores feature reduction** through KBest.
- **3rd and last iteration summarize the previous and fine tune** to get a final model to save.

The simplest pipeline resulted the best performing, so this was the final selection.



```
{'clf__criterion': 'entropy',
 'clf__max_depth': 200,
 'clf__max_features': 'auto',
 'clf__min_samples_leaf': 1,
 'clf__min_samples_split': 2,
 'preproc__cat_preproc__encoder__drop_invariant': False,
 'preproc__cat_preproc__encoder__min_samples_leaf': 10,
 'preproc__cat_preproc__encoder__smoothing': 0.01}
```

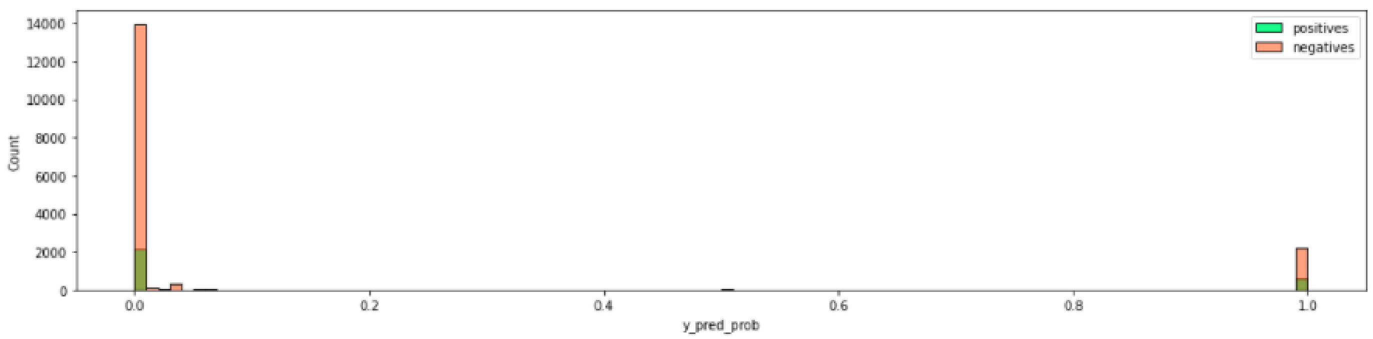
4.4. Visualization

Both ‘Explore Predicts’ notebook and front end .pys are part of the visualization block. In fact, front end is just a synthesis of ‘Explore Predicts’ notebook.

Before entering details let’s recap the project goals, as they were a) ad serving platforms data learning, and b) contact centre led prioritization. From there, we know that the key questions to visually answer are:

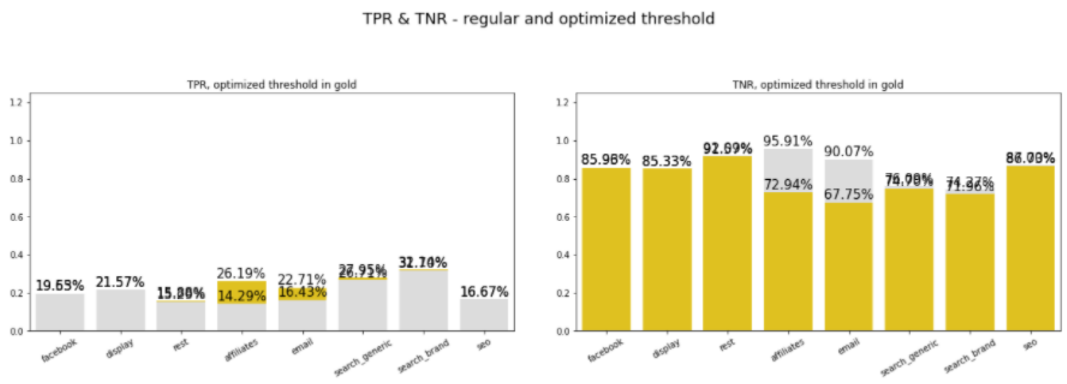
- How reliable is the model on Google and Facebook originated leads? As they are the big players where to use data for lead generation.
- Is the model failing because daring (too many positives) or conservative (too many negatives)? As agents should not focus on unprofitable leads.

For that purpose we plot the shape of positive and negative predictions on a histogram:

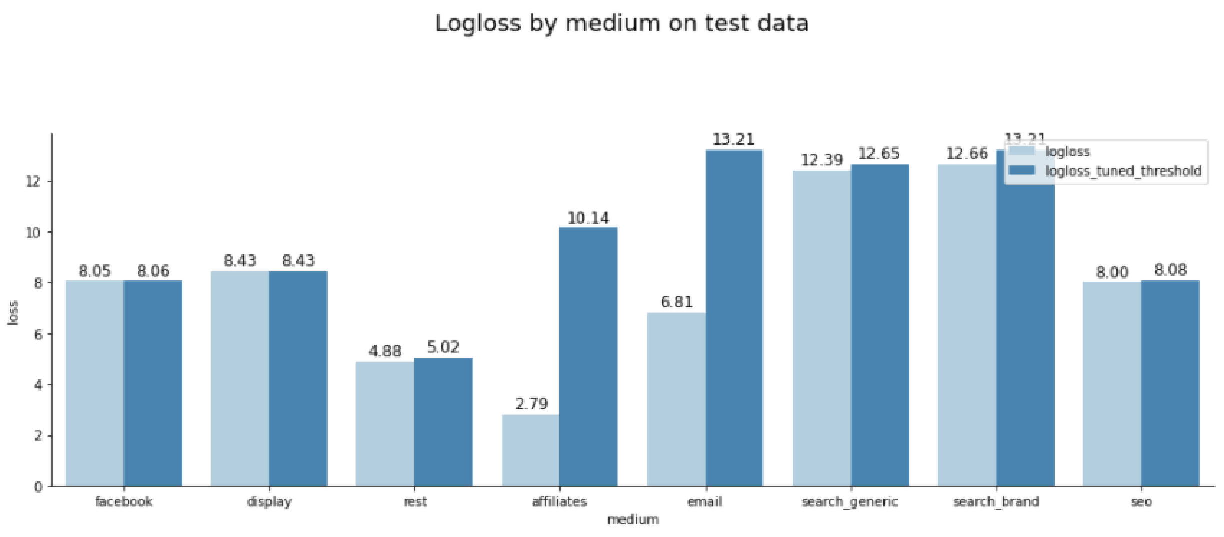


Seems like both **positive and negative predicted samples follow the same distribution**, but negatives are far more concentrated on almost 0 values. **Explore different thresholds** to adapt to this probabilities distribution can give more useful results, but for this case we are getting 0 as the best threshold, as an indicative of how the model struggles to detect positives.

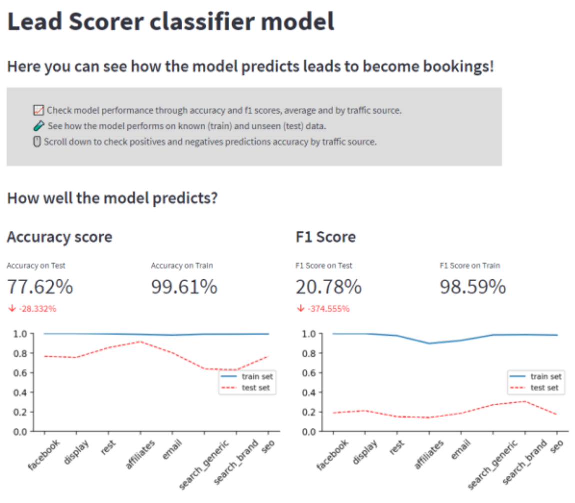
Thresholds can be explored also by medium, which also allows to answer how well the model works for specific Facebook and Google cases:



F1 score can be slightly improved when tuning the threshold, but still far from reliable results. Facebook and Google (search) cases are the ones were the model is more dare to predict positives, even at the expense of failing (TNR, logloss).



The **front end is designed for direction / managing roles on digital marketing and contact centre teams**, so it just easily shows the general model performance (accuracy & f1) by medium, adding a text reference about the metrics:



Lead Scorer classifier model

Here you can see how the model predicts leads to become bookings!

📈 Check model performance through accuracy and f1 scores, average and by traffic source.
🧪 See how the model performs on known (train) and unseen (test) data.
🖱 Scroll down to check positives and negatives predictions accuracy by traffic source.

How well the model predicts?

Accuracy score

Accuracy on Test
77.62%
↓ -28.332%

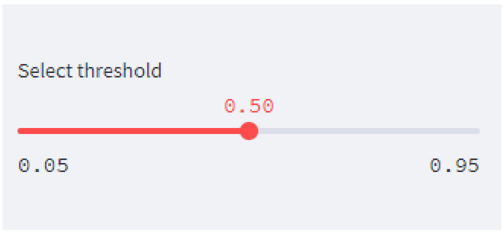
Accuracy on Train
99.61%

F1 Score

F1 Score on Test
20.78%
↓ -374.555%

F1 Score on Train
98.59%

It allows to explore different thresholds and check how this affects the metrics.



Select threshold

0.50

0.05 0.95

5. Conclusions:

1. Models couldn't detect positives because features are not good enough.
2. More severity on data cleaning can give better feature quality, and allow other encoding strategies, such as label encoders.
3. More feature engineering can help also on better feature quality, as most of them as they are too correlated between them, resulting in adding noise later to the models.
4. Also, other data sources should be considered, both on the main (with web data) and cookieless approaches.
5. Encoding ended up not being as decisive in the final models performance as expected at the beginning.
6. There was no real need on fine tuning model by model, as results weren't not really better than these same models base version, and computing cost was high.
7. Other approaches on how to deal with the predictions should be considered, perhaps by binning on clusters the outputs.
8. Once success is achieved in some model, the front end should focus on comparing the performance of those models, not only accuracy but only time performance.

6. How to run the project

6.1. Instructions for running the project:

- Clone or download the repository.
- Install requeriments.txt
- Download data files and place them on the same folder, on the 'data_files' subfolder. Please notice that files provided contain replicable data (csvs also produced on the project notebooks) but also non-replicable data (inputs for company and GA data).

*Note: GA Get Data notebook can't be replicated as it uses company resources and credentials.

Once everything is on the same place, notebooks can be run on the following order:

1. GA Clean Data: Clean and merge data from Google Anlaytics. This notebook outputs final GA data.
2. Analysis and Cleansing: Analyse, clean and transform all data. This notebook outputs pre-processed files ready to be used for modelling.
3. Model Testing (Main, Cookieless & SelectedFeatures): Play and understand different models. No outputs from here.
4. Model Optimizing: From a selected model based on previous notebook, fine tune hyperparams. This notebook outputs a) a model on a pkl file, and b) train and test predictions on csv files.
5. Explore Predictions: Analyse test predictions to understand model performance.
6. Front End: python script with the code for the front end (includes functions.py)

6.2. Front end user instructions

1. Open terminal and navigate to the project root folder.
2. Type 'streamlit run scorer_front.py' in the command line.
3. Wait for your web browser to open the app or copy the returned URL in your web browser.

You are ready to play with it!

7. Bibliography:

Brendan Hazs, ‘Representing Categorical Data with Target Encoding’:

https://brendanhasz.github.io/2019/03/04/target-encoding

Jason Brownlee, ‘How to Perform Feature Selection with Categorical Data’:

https://machinelearningmastery.com/feature-selection-with-categorical-data/

Jason Brownlee, ‘A Gentle Introduction to Threshold-Moving for Imbalanced Classification:

https://machinelearningmastery.com/threshold-moving-for-imbalanced-classification/

Aurélien Géron, ‘Aprende Machine Learning con Scikit-Learn, Keras y TensorFlow’.