MASTER'S FINAL PROJECT

AUTOMATIZATION OF MARKET RESEARCH DEPARTMENT:

MARKET RESEARCH HUB™

LLUÍS PELLEJÀ SOLDEVILA

MASTER IN DATA SCIENCE 2022/20223

Kschool

INDEX

1. INTRODUCTION

Welcome to my final master's thesis. In this document I am going to explain my project in the clearest possible way, so that any data scientist can replicate it without the complications that I have had when finishing it.

This project started as a small application that has evolved according to the needs of the company's Market Research department. These new needs have allowed the app to be very complete today, having many more modules than it had at the beginning, and solving problems in more than one department in the company.

I will separate this document into two parts, the first is where I explain the reasons and context why the development of this application was necessary, and why it made sense to do so.

In the second part I will explain the notebooks I have used to make the tests and the tests to make the machine learning model reach its maximum potential and above all I want to explain the data I have used to analyze how to make this model in the best possible way.

The third part of the document is the explanation of how the app works so that anyone with the same problem we had, can develop it or replicate it without facing the problems I had when doing it.

Having made this brief introduction, we can start with the Master's thesis report.

2. OBJECTIVES

This project was born with the main objective of saving and optimizing time for the Market Research department, in the following section I explain why it was necessary.

Once the project started, several problems arose that could be solved by adding more modules to the application, so the objective was not only to optimize time but also to help us make the best decisions about the sales and SDR departments.

The objectives that this application had to fulfill are the following:

- To optimize time in the search for SDR contacts.
- To increase the volume of contacts without the need to increase the number of staff.
- To be able to measure and analyze the metrics of the SDR team in real time and in the same tab.
- Be able to listen to the most important calls of each SDR in a quick way
- To be able to show updated metrics to the SDRs with access control so that they do not see sensitive information.

In short, the main purpose of this application was to automate processes so that the Market Research department could dedicate its efforts to strategic reasons and leave aside the more operational and manual actions with the main objective of providing more value to the company.

3. CONTEXT

In this part of the dissertation report, I would like to give a bit of context as to why it was important to do this project, as I believe that this way the problem it solves and, above all, the value it brings will be better understood.

First of all we will separate the app modules into three parts:

- The duplicate check
- The one for generating company info
- The part of the KPIs and calls

Duplicate Check:
In the section on checking duplicates, what the programme does, roughly speaking and without going into detail, is that it examines the companies that we pass it to check and checks whether they are clients or are in "pilot test" within the ERP[1].

Also checks whether they are in the CRM[2] (Pipedrive), in the case that they are in the CRM and have won or open business, the app understands that it is a company that we cannot call as it was already won in its day, but if the deal is in the CRM and is lost and that more than 90 days have passed, it can try to call it again.

Also in the module to check duplicates you can merge in one click the duplicate companies in the CRM, this part is explained in more detail in section 5 a), where I only talk about this module in question.

Before we had the application, what we used to do was to generate an excel file with all the clients every week, to keep it updated and with an excel formula we crossed it with the companies we wanted to prospect and we kept deleting them.

Once the clients were deleted what we did was that we went company by company checking if in the CRM they were open, won or lost, in the case of lost clients we had to look at the annotations, the reason and the time they had been lost, which meant a lot of time.

With the application we not only save the time of doing it by hand, but while the programme is working we can do other things that add more value.

With the adoption of the programme we have gone from taking 30-40 seconds per company to being able to check each company in 2 seconds, which means that we can now make 20 times the contacts that we made by hand and make them better because the human factor is no longer present, so the contacts that reach sales are checked and are of quality.

---

[1] **ERP**: is the system used by the company to store internal customer information, incidents, invoicing and everything to do with them.

[2] **CRM:** is where the sales department has all the contacts they are prospecting, it helps us to organize calls, meetings and have all the information of all the prospects in one place. In our case we use Pipedrive.

Generate info context:
In the module of generating the information two parts are executed, the first one is the same as the previous one, this was done because sometimes we only want to check if we have repeats or not and others we want to see if there are repeats and those that remain we want to know their information, for that reason we put these two modules. The second part, as I said now, is to generate the contact information of the companies.

Also in the module to generate the information you can find the e-mails of the contact persons that you have in the bases, through the link of the company or their name and surname (with the link of the company), about this functionality I will talk more in detail in section 5 b).

This second part of the module is also complicated because what it does is to collect all available contact information such as: postal address, telephone numbers, e-mails, website, contact persons, invoicing and employees through the companies' VAT number. All this information is stored in an excel file and the user can download it in CSV format.

Previously this information was extracted by hand from databases or the internet, doing the companies one by one and copying and pasting the information into an excel spreadsheet. This meant that the department could only do this, as you can imagine it was a tedious and mentally tiring job.

In addition, before the programme, we used to find the fleet size[3] by searching the websites, which made it even more time consuming to generate a lot of contacts in a short period of time.

With the implementation of the app we have gone from being able to manage 150-200 contacts a week to being able to do it in 1 hour, so now no matter how many contacts there are or how many sales people the company has, we can supply the sales department without the need to incorporate more people to make databases manually and without sense.

KPI and SDR calls:
The KPIs of the SDR and sales department are controlled by Market Research, this is because market research has a broader global view than sales so it can see beyond each profile and analyze it properly.

In the application there are different modules to evaluate the KPIs but in the end they all lead to the same conclusions. In the most important KPI module we measure the SDR indicators individually, collectively and save the previous month's KPIs for comparison in the same app.

[3] The fleet size of the companies is of interest to us as Movertis is dedicated to commercialize fleet management software to companies that have vehicles, these companies are billed more or less depending on the number of vehicles they have, so it is important to segment the companies by the size of turnover they are likely to carry.

KPIs come from two main sources: CRM (pipedrive), the virtual switchboard[4] (Aircall). Both softwares stores the department's data and sends it to the app to display it in the most convenient way for us.

Previously, in order to see the same KPIs I had to run a report in Pipedrive, which took about 15 seconds to load and then I had to filter by user if I wanted to see the segmented indicators or filter by team if I wanted to see them grouped.

In the case of Aircall what I was doing was filtering the calls and then profile by profile counting them by hand and then seeing the approximate average durations and calls made by hour.

With the app this process becomes a matter of clicking a button and seeing it in a matter of 5-10 seconds, filtered, sorted and with even more data applied to our use case.

With this implementation we can make decisions in real time as the app shows us the KPIs updated every minute and a half.

We have also incorporated a section where with just one click you can listen to calls that have lasted more than 5 minutes and see in a matter of seconds where you can improve to gain more customers.

Conclusion of the context

As I have explained, making this application not only solved the issue of my TFM but in the company where I am this application, despite not being very complex, has solved the life of a whole team of Market research that has gone from being almost all the working day chopping excels and doing work by hand, to automate it and focus on what really matters, the sales strategy and training of all SDR profiles, in order to generate more business opportunities and the company to grow faster.

[4] The virtual switchboard is the programme from which all the SDR profiles make the calls, this allows us to centralize all the telephones in the same place and above all to centralize all the data generated by the telephone, so that the extraction, cleaning and analysis of the data becomes much simpler.

4. DATA USED

In this part of my master's thesis report I will break down the data I have worked with and the two test notebooks I have used in order to get the most out of one of the fundamental parts of the information generation module, the machine learning model.

To do this we will make a small explanation of the notebooks and especially of the data that I have used to do it. I will also present the datasets that you can use to test the modules, since obtaining data in this project is a bit peculiar.

a. NOTEBOOKS USED

In this project I have used two notebooks to make tests and trials in order to get the most out of the training data, the notebooks can be found in the repository in the notebooks folder or in this GitHub link, the notebooks are as follows:

- analysis_of_input_table.ipynb
- model_tests_fleet_size.ipynb

Both notebooks have their own particular purpose and were both used to determine the Kedro[5] pipelines (one for the preparation of the table for input into the model, and the other to determine which model was the most suitable for the use case on the table).

Analysis of input table notebook:
Although the notebook explains the reasons for each action, I believe it is necessary to do so in a synthesized form in this document and if any doubts remain, it can be consulted for more details.

The purpose of this notebook is to analyze the dataset we have of all the historical companies (with all their information) that we have surveyed since January 2022.

From this file we want to draw several conclusions, first of all, which are the variables that most affect the fleet size, since, if it is the value we want to predict, we need to know the independent variables that affect the value of the dependent variable, in this case the fleet size.

In this case we make different assumptions: the sector of the company, the annual sales and the employees will be important and almost certain independent variables.

At the same time I want to analyze the dataset to see if the variables: has an email (yes or no), has a website (yes or no), what type of phone it has (mobile or landline) and what part of the territory the company is from are important enough independent variables to add to the machine learning model, the notebook explains the reasons why I thought at the time that they might influence our dependent variable.

[5] **Kedro:** It is the technology I used to create the project, thanks to it I have been able to keep my folder system as clean and tidy as possible, I have also used it to generate the Pipelines of the transformations that were made in the tables in all the processes.

The conclusion of this notebook is that the variables that are best related to fleet size are the expected variables (sector, annual sales and employees), and it concludes that the other variables we have tested cannot be completely ruled out even though they do not have the same relationship with the dependent variable.

In short, the notebook concludes that the models should be tested with all variables to understand which one works best and with which variable. Another conclusion I drew from the notebook was that the sector variable had to be narrowed down to 5 possibilities[6], as the ones that fall under the label "other" are those for which there is not as much data to make reliable predictions.

Model test fleet size:
In this notebook what we want to compare are different things, the first one is to see which model is the best to predict the type of dependent variable we have, I also want to check which dependent variables are better to test these models and also compare the machine learning models with the logic[7] that we applied before the app was developed.

The models we are going to test are two types of models: regression models and classification models because the dependent variable can be expressed in two ways: as a numerical value and as a range between two values (this is how we express it in the CRM), this is the example:

Range[8]	Numerical values[9]	Range	Numerical values
1 - 5	2.5	**51 - 100**	75
6 - 10	7.5	**101 - 200**	250
11 - 20	15	**201 - 500**	325
21 - 30	25	**> 500**	500
31 - 50	40		

Table 1: relationship between fleet size ranges and their numerical values
Lluís Pellejà 2023

[6] The final sectors I decided to incorporate were: Transport, Coaches, Distribution and Technical Services. The others were included in the same sector due to the lack of data with which to train the machine learning model.

[7] Before developing the machine learning model and before automating the process in the company we took into account the turnover when assigning fleet sizes, this logic is explained in the notebook but, what we did was that we assigned in each sector a value of turnover per vehicle and it used to go well, to understand how well or bad the models go we will compare it to our "handmade" model.

[8] The fleet size ranges allow us to easily classify companies into different ranges and thus understand what size of company we are targeting.

[9] To represent the fleet size as a numerical value, we take both values of the range and apply an average to them. I understand that it will not be the exact fleet value of the companies but it will give us an approximate idea of what range the analyzed company is in.

In the notebook what we try to test is first of all which type of model is best for the types of data we have, what we see when testing the regression models is that despite passing the fleet sizes to numerical values, they don't work quite right.

This result is due to the fact that the independent variables and the dependent variable do not have a strict linear relationship, so in a first set of tests we already see that the classification models will be the best positioned to handle this type of data.

After testing the classification models I saw that they were the correct models, I simply passed the encoder to the dependent variable of the training set and it matched perfectly, so now that I had found the type of model I needed I had to test the three models that define classification models par excellence: Decision tree, K-nearest neighbors and the Random Forest.

Once I was clear about the models, I prepared the tests with all the combinations of variables that we decided on in the previous notebook and I ended up deciding on the Random Forest, as it was the one that showed the best performance in equal hyperparameters.

Once the machine learning model was chosen I had to find the optimal hyperparameters for my data, for this I used the GridSearchCV library to make the necessary iterations to find the hyperparameters that best fit my data and made the best prediction so I could choose the hyperparameters for my model.

In order to get my doubts out of the way and explore all the options, I tested the GridSearchCV library in all the classification models and it turned out that in the Random Forest model it was still the one that scored the highest, so I decided to use this one.

As an advancement to the testing since we have been using this technique to predict fleet size we have received less complaints from the SDR department on this part so after 3 months of implementing this model on fleet size I can say that it is working very well.

The conclusions we draw from this notebook are that the best decision based on the data and now on the results was to choose the Random Forest model as we did at the time, the notebook helped us make that decision and it was the right one.

b. DATA TO ANALYZE THE INDEPENDENT AND DEPENDENT VARIABLES AND TRAIN THE MODEL.

The data to analyze the independent variables and the dependent variable is the file called *raw_input_table.xlsx*[10]. This file contains all the companies prospected by the SDR team since January 2022, with a total of 5,051 company records with the information filled in and most importantly the fleet size.

[10] The shared file is confidential and can only be seen by the people with whom it has been shared, and its dissemination is completely forbidden.

This data set has 27 columns but we are not going to use all of them, we will keep the following:

Column name[11]	Value type	Column name	Value type
Name	string	Fleet size	string
Sector	string	Region	string
CNAE code[12]	int	Phone	int
Annual sales	int	Mail	string
Employees	int	Website	string
Mean fleet size	int		

Table 2: columns of file raw_input_table.xlsx and their value type
Lluís Pellejà 2023

In the file we find everything we need to analyze the impact of all the independent variables and to reach interesting conclusions. This file keeps growing week by week as we add more companies to the database, so the model is trained every week with more data.

The aim is to reach a point where all sectors can be independent of each other and where the base is so broad that there is very little wrong with the prediction. The company believes that we will be able to reach the target by the end of this year, which is why the model is constantly changing and evolving on a weekly basis.

c. DATA TO CHECK THE FUNCTIONING OF EACH MODULE

The files I have shared to test the two modules of the app are focused on companies in the sectors that interest us. You will see that I have prepared several files so that you can see that the app is agile and all the commands work.

Duplicate comprobation file:
To test the duplicate management module I have added a file called "*check_duplicates.xlsx*[13]", in this file you will find 15 companies, which when passing them through the duplicate check should be 11 (tested on 07/10/2023, as the result may vary depending on what we have imported into the CRM during the last few days).

[11] In the file, columns are written in Spanish, this is due the fact that the file is used by the company and everybody has to understand it.

[12] CNAE codes are in Spain the number code to differentiate each sector and activity from others that could be similar.

[13] The shared file is confidential and can only be seen by the people with whom it has been shared, and its dissemination is completely forbidden.

In the file of the good companies there should be two extra columns, one with the reason for the loss of the companies that have been imported, and the days that they have been lost.

The process for testing the information generation module is very simple:

1. *You enter the app*
2. *In the side panel select "Check for duplicates".*
3. *Drag the file over the importer*
4. *Click on the button "Start scanning".*
5. *Wait for the results to be displayed and ready to download*

Generate the information file:
To test the module to generate the information of the companies I have attached a file called "*generate _the_information.xlsx*[14]" This file is 14 companies that we want to prospect so, we need to see if there are any repeated and if they are not we want to contact them so we will need their information.

The process for testing the information generation module is very simple:

1. *You enter the app*
2. *In the side panel select "Generate information".*
3. *Drag the file over the importer*
4. *Click on the button "Start scanning".*
5. *Wait for the results to be displayed*[15] *and ready to download*

Generate mail with name, last name and link:
To test the functionality to generate the mails of people with their name, surname and link of the company where they work, I have added a file called "*get_mails_name_lastname_link.xlsx*[16]", with this file you can test the module without problems.

You will see that it does not have all the mails of the people, it is the most common, but this module helps us to improve the contactability in companies that are very hermetic.

The process for testing the information generation module is very simple:

1. *You enter the app*
2. *In the side panel select "Generate information".*
3. *Select the 2nd tab*
4. *Choose the first option*
5. *Drag the file over the importer*

[14] The shared file is confidential and can only be seen by the people with whom it has been shared, and its dissemination is completely forbidden.

[15] The results only show the companies that can be prospected, so if any of the companies have been imported into the CRM during the course of these weeks, they will not appear in the results.

[16] The shared file is confidential and can only be seen by the people with whom it has been shared, and its dissemination is completely forbidden.

6. *Click on the button "start generating mails".*
7. *Wait for the results to be displayed and ready to download*

Generate mail with link:
To test this module I have added a document called "*get_mail_link.xlsx*[17]" that contains the urls of different companies, so that with the module you can find the mails of the most important people in the company.

This is the module that is less used because it returns all the data mixed and needs to be polished but it is good to see the size of the company and the people who work there, or get an idea of what pattern their mails follow.

The process for testing the information generation module is very simple:

1. *You enter the app*
2. *In the side panel select "Generate information".*
3. *Select the 2nd tab*
4. *Choose the second option*
5. *Drag the file over the importer*
6. *Click on the button "start generating mails".*
7. *Wait for the results to be displayed and ready to download*

Get all the mails from a link:
To test this module you only need one link, as it is focused on searching the mails from a single link and sorting them in the same application so you can copy the one you are most interested in, some suggestions are the ones in the previous file, as it will show you the results sorted in a table.

The process for testing the information generation module is very simple:

1. *You enter the app*
2. *In the side panel select "Generate information".*
3. *Select the 3rd tab*
4. *Write the URL in the textbox*
5. *Click enter.*
6. *Wait for the results to be displayed*

[17] The shared file is confidential and can only be seen by the people with whom it has been shared, and its dissemination is completely forbidden.

5. THE FUNCTIONING OF THE APP

In this module I will finish explaining how the different modules of the app work, for this I have grouped the modules in three parts, as several modules are used for similar things and their functioning is quite similar.

To explain them I have divided them into the three main modules ("Repeat management", "Duplicate review" and "KPIs"). So let's explain them in more detail.

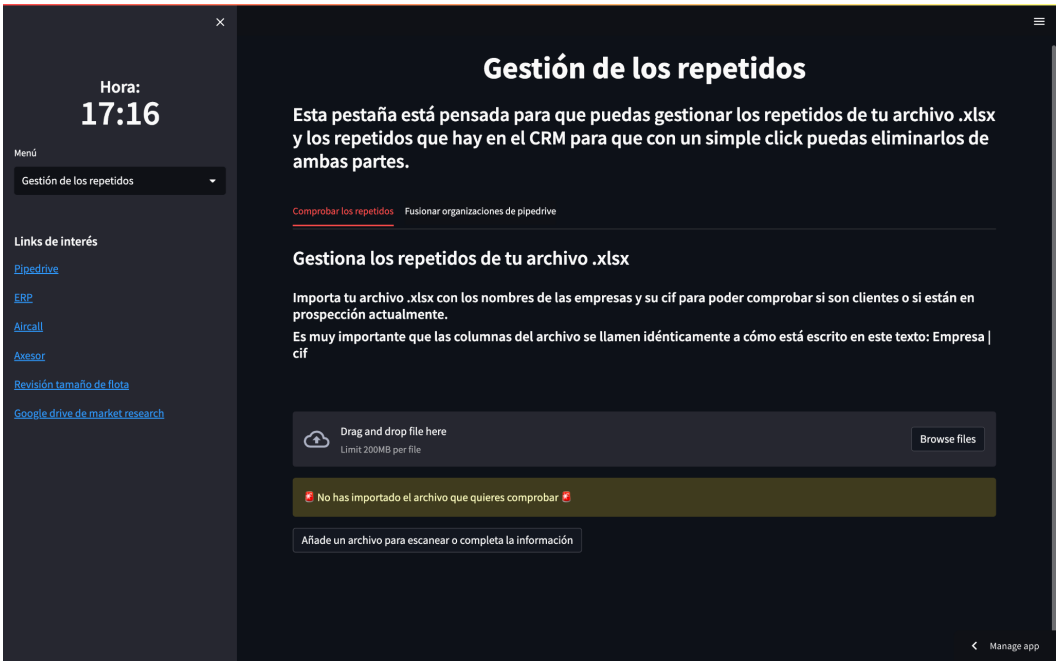
a. DUPLICATE COMPROBATION

The first module I would like to explain is the duplicate review function. This module was born from the need to streamline manual processes and, above all, to optimize the time of the people working in Market Research.

Previously this management took a long time to carry out and the main problem was that many companies that were repeated or were already in the CRM passed the filter, it was a normal thing, in the end the human factor was present and you could not expect that a person doing a repetitive task was concentrated doing the same repetitive task.

In order to put an end to these human errors we programmed a script that I am going to explain below in a simple way since in the code there are all the necessary annotations to be able to replicate it without any problem.

Functioning of duplicate comprobation module:
The first thing you find when you enter the module is the following:



Image[18] 1: Duplicate comprobation module
Lluís Pellejà 2023

[18] All the images have the text in Spanish because the app is used by different users and the aim is for everyone to understand it perfectly.

As we can see in *Image 1* it is a module that apparently is very simple, we have a small explanation at the beginning and a file importer with a Warning saying that we have not imported any file to scan.

Once the file containing the companies to be scanned is imported, the module looks like this:

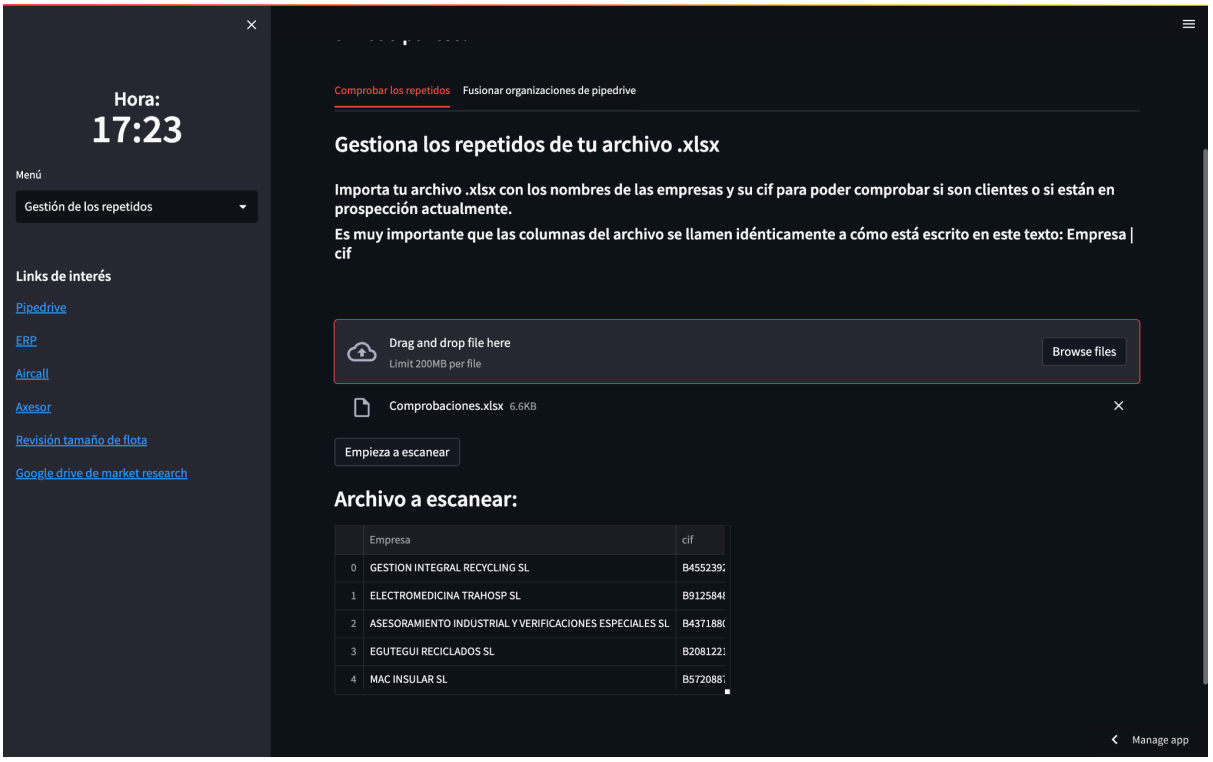


Image 2: Duplicate comprobation module with the imported file
Lluís Pellejà 2023

As we can see in the image the button changes text and the imported file is shown to preview it before starting to scan, once we click on the button there are several processes that start to happen.

The first process that happened while the file was uploading was that the excel that was uploaded to the platform was sent to an AWS S3 bucket, so that the file cannot be lost again, using the function called "*send_file_aws()*" that is in the "*app.py*" file.

Once everything is ready and the file is well imported, click on the button and the first Kedro pipeline called "dc" (duplicate comprobation) is activated.

The pipeline works as follows:

1. *File download:* Import the file from S3 to trigger the whole process.

2. *Customer validation:* Imports the list of active customers directly from the ERP and separates the companies that are customers and those that are not into two Datasets with the function "*customer_examination()*".

3. *Validation in the CRM:* It crosses the companies that it has verified that they are not customers with the CRM database to find the matches, in the case that there is a

match it looks to see if it can be uploaded or not and in the case that it is yes, it adds the relevant information for future evaluation. The companies are divided into two datasets: "good" and "bad" companies. This process is done by executing the function called "*crm_info()*".

4. *Concatenate invalid companies:* The last step is the concatenation of the DataSet that has been eliminated from the part of the clients and from the part of the CRM validation.

The structure is as follows:

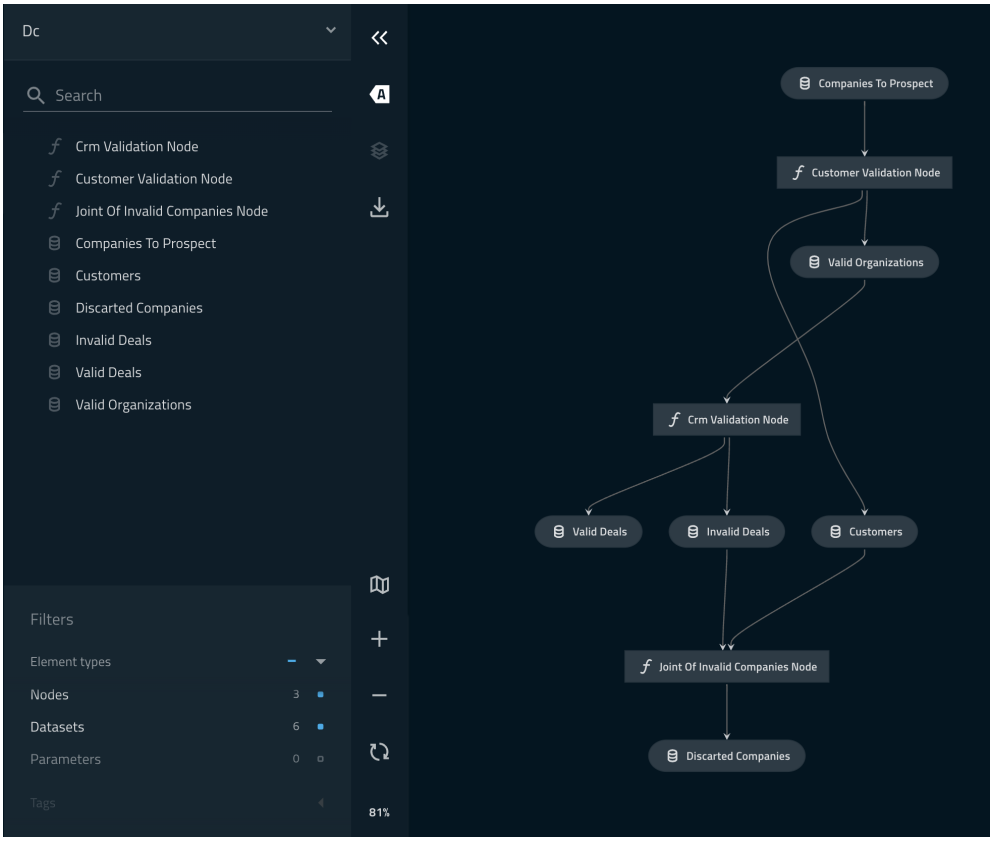


Image 3: Duplicate comprobation pipeline
Lluís Pellejà 2023

The complete functioning of the pipeline is explained in the code but as we can see there are simply two steps that have two very well defined functions, the checking of customers and the checking of non-customers.

As input we have the user's table that after saving it in S3[19] the pipeline uses it to activate itself and at the end of the pipeline we have two outputs: the valid companies and the invalid companies, in this case the valid ones are shown to the user.

[19] The "bad" companies are stored in S3 in case someone from the department wants to check them.

Once the process is finished, the application displays the table of companies that have passed the filter, with some metrics just below the results, this is how this screen would look like:

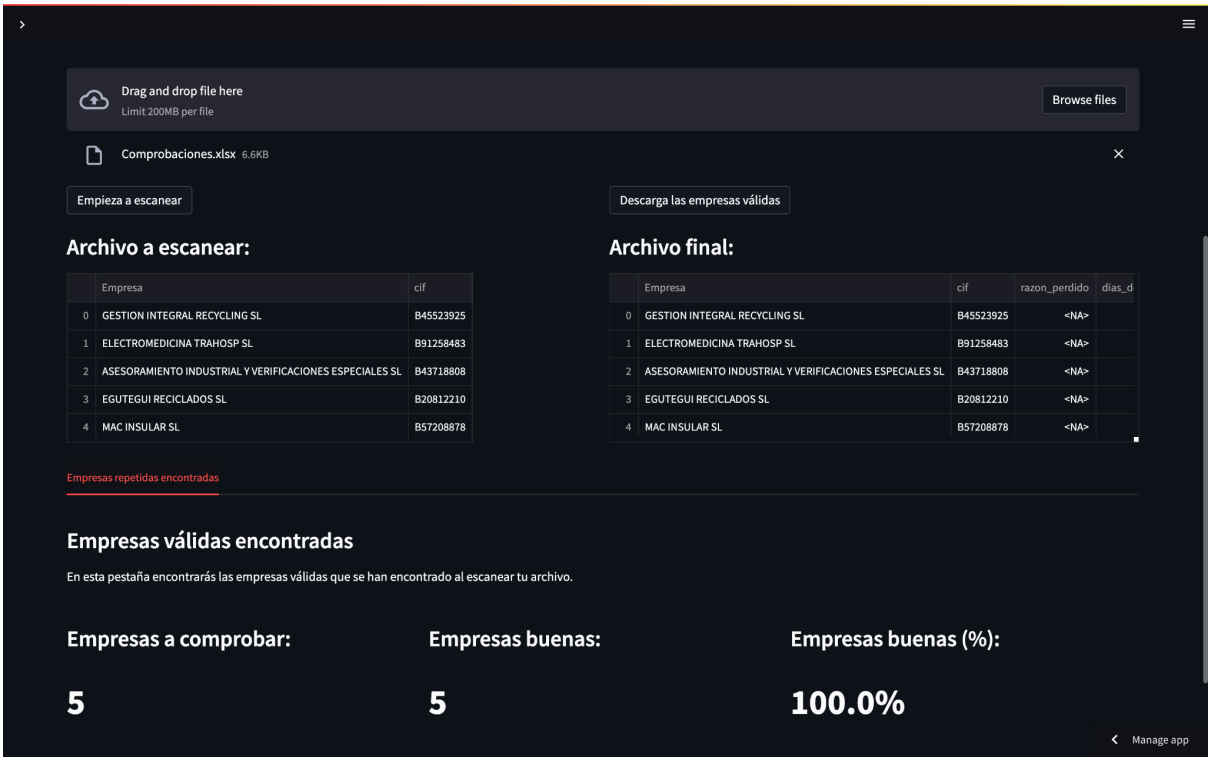


Image 4: Duplicate comprobation pipeline finalized in the app
Lluís Pellejà 2023

As we can see in image 3, the system itself tells us the percentage of companies that have passed the filter to give us a quick idea of whether we need more contacts or whether we can work with them.

We can also see that the application displays a download button above the result table, this button allows us to download the result table in CSV format to use it for whatever the user wants.

Functioning of merging duplicates in Pipedrive:
This module has a very simple system as it does not activate any pipeline in Kedro, it is just a function that is explained in the code in a very detailed way.

The first part is identical to the duplicate check as it only contains a file importer and a button to initialize the function.

The function that is activated when touching the button is called "merge_org()", in short this function what it does is to look for how many identical businesses there are and to merge them with the most recent of all of them saving the information of all the businesses in only one, avoiding the loss of information and the saving of space in the CRM.

We know which is the most recent because the IDs in Pipedrive are created in order so that by sorting the list of identical company IDs in descending order, we know that the first one is the newest, and therefore the one with the most up to date information.

b. GENERATE COMPANY INFO

When we enter the module to complete the information, as users we find a screen very similar to the previous one, there is a small explanation at the beginning and then the importer with a Warning that indicates to the user that he/she has to import a file.

This is what the initial screen looks like:

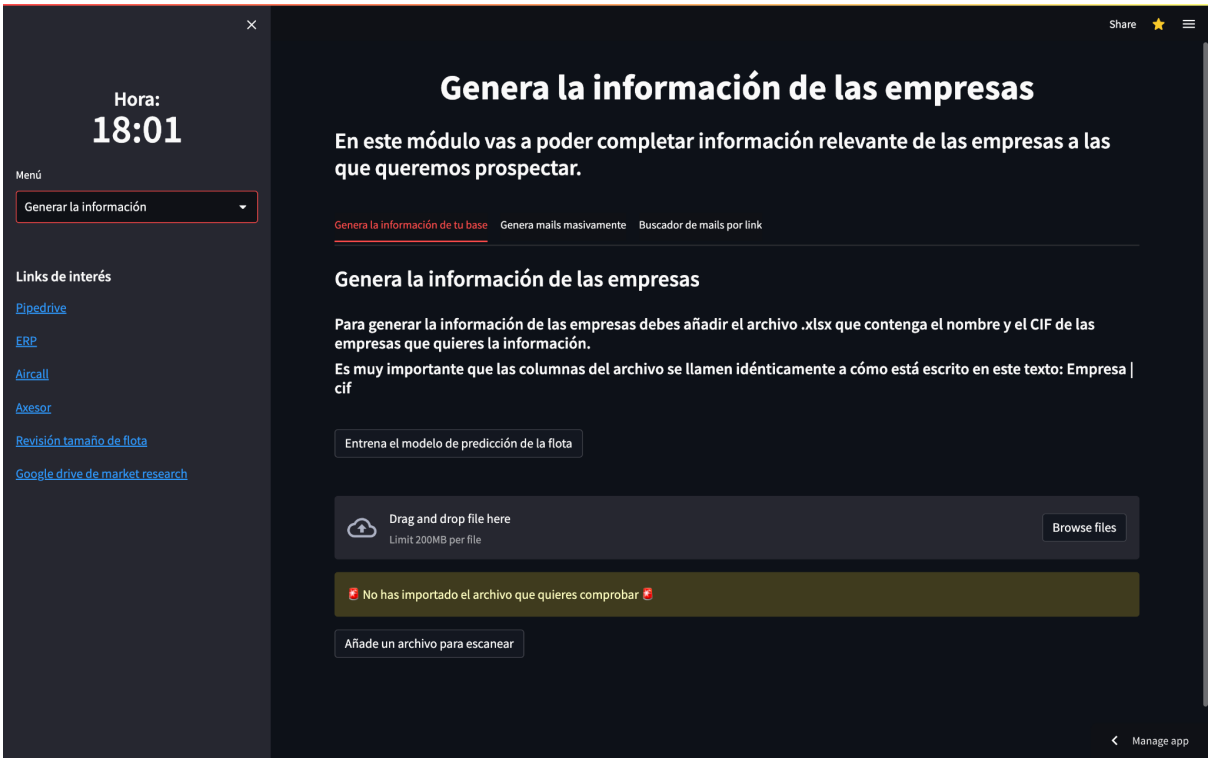


Image 5: Get company information module
Lluís Pellejà 2023

In this module, when the user imports the file, it does exactly the same as in the previous module, it sends it to the AWS S3 and prepares the activation button of the corresponding pipeline.

In this case not only one pipeline is activated but 3 different pipelines are activated:

- The one I have explained above for duplicate checking.
- The pipeline to complete the information
- The pipeline that downloads the machine learning model and applies it to the data collected in the previous step.

I will explain in this document how the 3 pipelines work, all of them are very well explained in the code with their respective messages.

Once the user presses the button, the first pipeline that is activated is the check for duplicates, which is already explained in the previous point.

Once the first pipeline has finished executing, the valid companies become the input of the next pipeline: the one to generate the information.

This pipeline has only one step, which is the function that goes through the table of valid companies and searches through different databases for the contact information. This function is called "*get_info_company()*" and is explained with comments in the code.

This is what the pipeline looks like when completing the information:

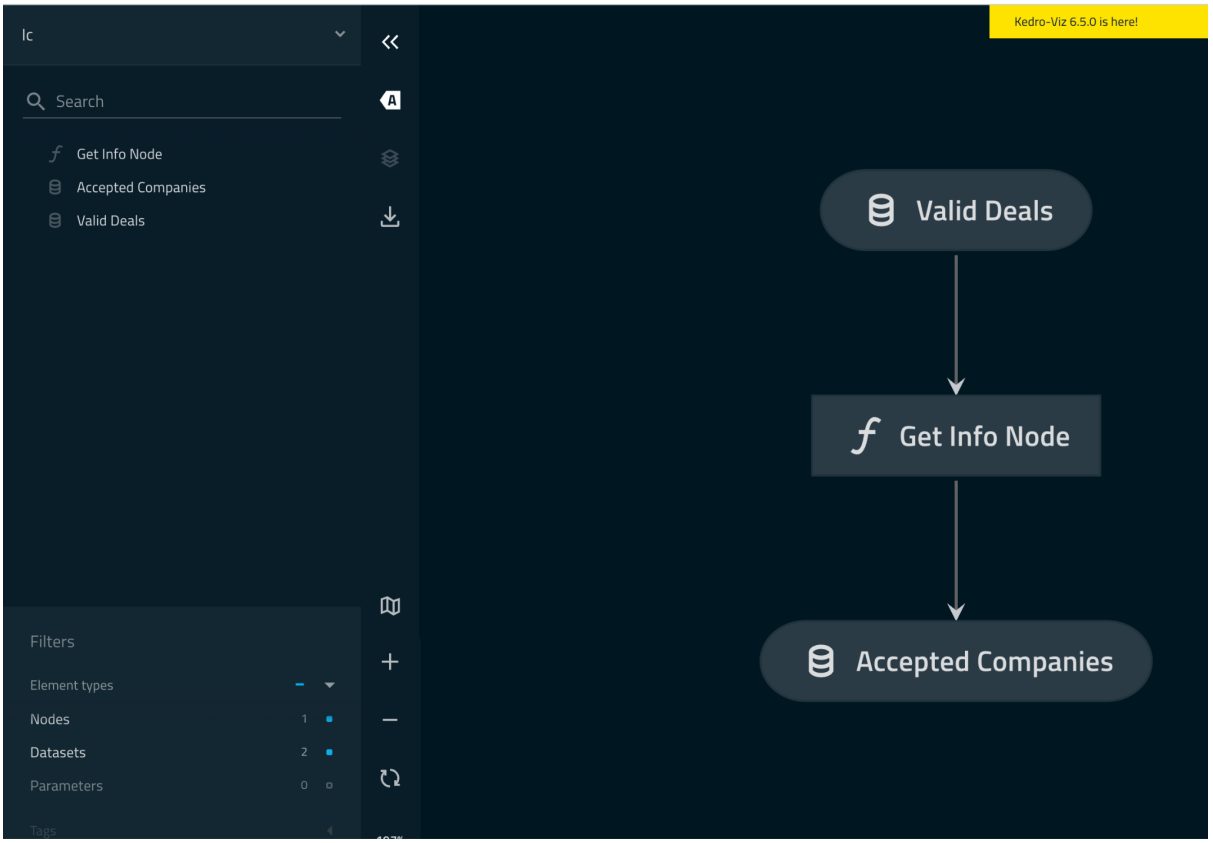


Image 6: Get company information node
Lluís Pellejà 2023

As we can see in the previous image this pipeline has only one input and only one output, this last one will be the table with all the information of the companies, and also the table that will enter in the next and last node: *the fleet_value_attributes*.

This pipeline is more complex than the previous one, in this case there are three quite important steps to finish preparing the table for the user.

The first step is to take the table that came out of the previous pipeline and divide it into the companies that have all the information to go through the model and those that do not, we are consulting the information of many companies and it is very likely that in the database

there are companies that do not appear in the register so it will not complete the information for all of them.

This separation will be done with the function called "*prepare_data()*".

Once we have separated in two Datasets the companies that have enough information to go through the model and those that do not, we can continue to the next step.

The next step is to download the trained model that predicts the fleet size and import the table that came out of the previous function with all the information needed to try to predict the fleet size.

With the function called "*prediction_of_fleet_and_value()*", we predict the possible fleet that each company in the table has.

Once the companies have passed through the model we decode the result and add it to the table to start creating the final table. In this final table we add the companies that have not gone through the model and add the necessary columns.

Once this pipeline has finished its execution, the result table is shown on the screen.

This is what the complete pipeline of this module looks like:

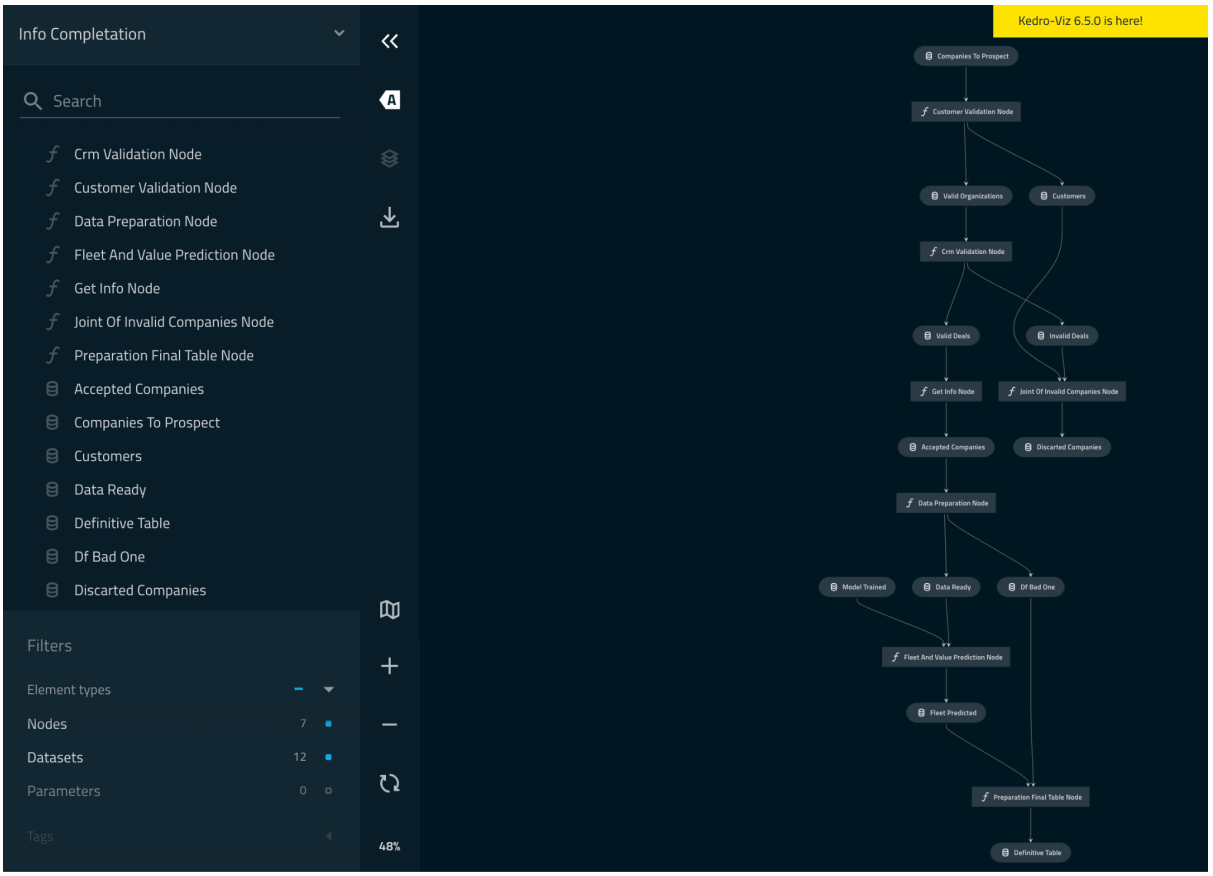


Image 7: Get company information all node
Lluís Pellejà 2023

This is the screen the user sees when the pipeline has been fully executed:

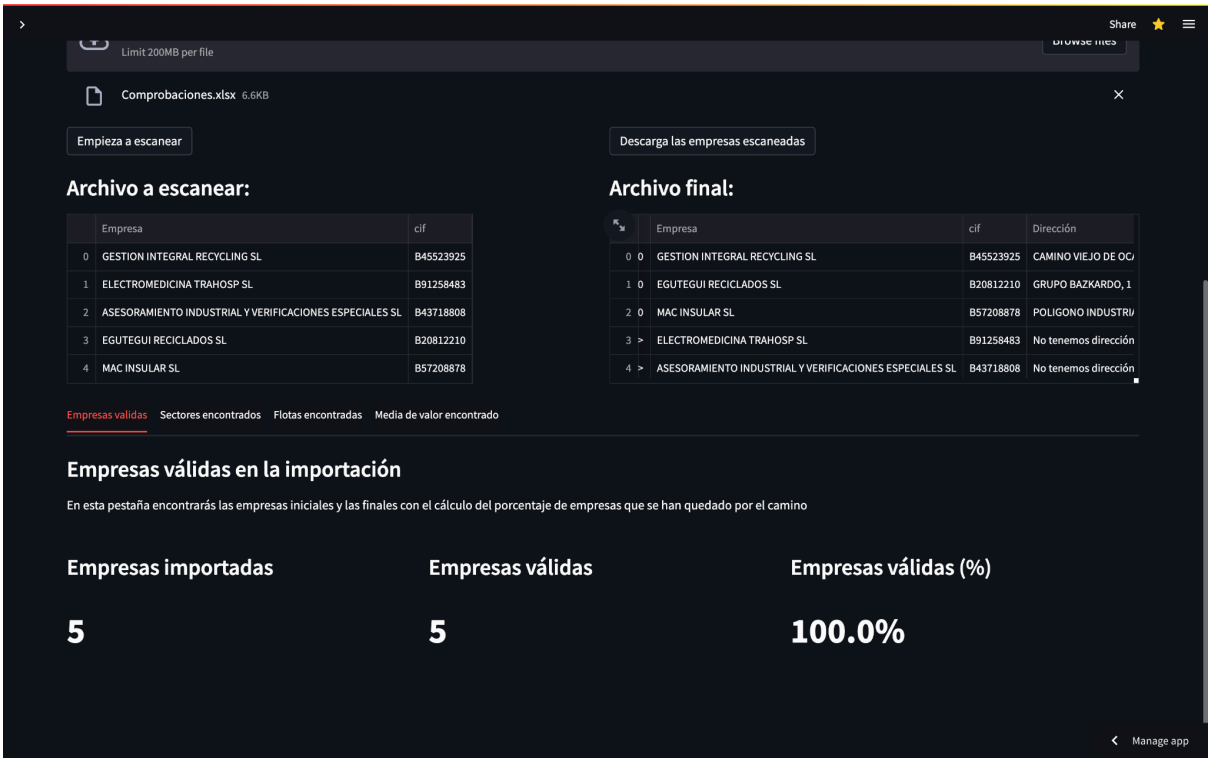


Image 8: Get company information pipeline completed
Lluís Pellejà 2023

Once the information has been found and the pipeline has been completed, some KPIs are displayed at the bottom of the screen so that we can take a quick look before downloading the file.

As it was said before, the app lets us download the file in CSV format so that the user can do whatever he wants with it.

Train the machine learning model:
This module is very simple as it only activates a Kedro pipedrive which is very simple, in the section to generate the information of the companies there is a button that says "*Train the prediction model of the fleet*" by clicking on this button the user activates the pipeline called "*model_training*" which looks like this:

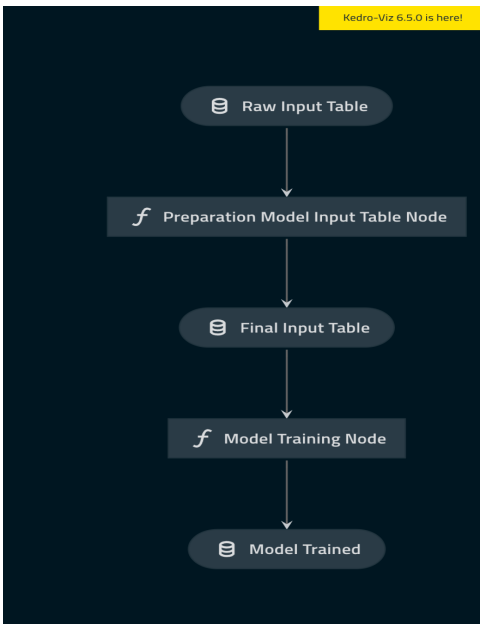


Image 9:
Train machine learning model pipeline
Lluís Pellejà 2023

As we can see in the image, all the pipeline does is download the previously saved S3 table called "*raw_input_table*" where the history of

companies that have been prospected in SDR is stored and transforms it to pass it through the model with the function "*preparation_model_input_table()*" which is explained step by step in the code.

Once the table is prepared, the training begins, which is carried out through the "*model_training()*" function, which is responsible for training the model with the table already processed and saving it in the AWS S3 so that in the future the user can predict the fleet size of other companies with this model.

Get mails with name, lastname and link:
This functionality has a very simple operation since it takes the dataframe with the columns Name, Surname and Link of the company and with a simple call to an api it gets the names in each row of the DataFrame that the user imports.

This is done through the function called "get_email_name_last_name_link()" which goes through the DataFrame looking for each of the mails that you indicate and in the case of not finding it, it puts an empty element so that the function does not stop with an error.

The operation is very similar to the other modules since it has a small explanation with the basic instructions and an importer with a button so that the user can import his file and activate the function.

Once the results are available the result is displayed on screen and can be downloaded in CSV format so that the user can do with the resulting table whatever he wants.

In the case that the user only has the data of the web page in the drop-down select the second option and the same interface comes out but when pressing the button the function that is executed is called "*get_email_link()*", this one what it does is that for each link it saves in a table all the personal emails that it finds so that the user can choose better.

Get mails with one link:
To finish the module of searching for information about the companies, the app presents a functionality that allows the user to provide a link and the programme returns in a DataFrame the people, the position and the email address of the company.

To do this the user has to enter the link in a text box found in the module and press enter, pressing it activates a function called "*get_emails_one_link()*", this what it does is qeu makes an API request to the service and returns in an orderly manner all the values.

This functionality is very useful when the user needs to find the email of a specific person or simply needs to find a contact person of a specific company.

c. KPI AND DASHBOARDS MODULES

The remaining modules of the application are modules that focus on measuring the KPIs of the SDR and support team and they all use the same methods as all the information is extracted in the same way and will be explained in detail in this document.

The sub-modules to be explained in this document are:

- SDR KPIs individualized this month
- Group SDR KPIs this month
- Group SDR KPIs last month
- Important SDR calls

These are the most important modules so almost all of them have the necessary functions to extract all interesting KPIs for Market Research.

SDR KPIs individualized this month:

In this module, the app shows the most relevant KPIs of the month segmented by each SDR profile in the team, depending on the profiles in the team.

In the code you can see at the beginning of the module called "KPI - SDR Team" (from line 1010 to line 1076) that all the variables that will be used during the whole module are defined so that when changing the users the program administrator can do it in a simpler way.

The variables that the program uses to collect the data of the calls from the virtual PBX and the CRM data are the following:

Variable[20]	Usage
Users	This is the list of user names to be used in the frontend.
Colours	The color assigned to each user for the visualizations.
Aircall IDs	The IDs of each user in the software of the electronic switchboard.
Won filter ID (current month)	The IDs of the CRM filters containing the business won by each SDR.
Open filter ID (current month)	The IDs of the CRM filters that contain the businesses that keep each SDR open.
Lost filter ID (current month)	The IDs of the CRM filters containing the business that each SDR has lost.
Inactive deals filter ID (current month)	The IDs of the CRM filters containing businesses that have not been active for more than 3 weeks.

Table 3: variables used in SDR KPIs individualized current month module
Lluís Pellejà 2023

[20] These variables are the ones used in this part of the KPI module.

As we can see in the table, the user has these variables to extract all the necessary information from the module, in the following key points we can see which functions are used to extract the information and save it so that it is visible in the app.

- ***status_check:***
  This function takes as input values two lists already defined in the previously mentioned variables: the users and the aircall IDs, with these values the function makes a request to the API for each user in order to get their current status on the phone, with the data it creates a dictionary that as a key has the names of the users and as values the status sent by the API, which returns it for manipulation.
- ***get_calls_time_total_calls:***
  This function takes as input values the list of users' aircall IDs, once executed it makes the request for each user simultaneously, which saves time and optimizes resources. What the request does is to ask the Aircall API for outgoing calls from a specific user during the day in which the function is being executed.

  The function goes through the calls that the API has returned and counts those that last more than 1 minute and saves their duration, and also saves the value of the total calls.

  Finally the function saves how many calls in total the user has made, how many calls are longer than one minute and with the durations of these last ones it makes an average to see the average duration of the calls longer than one minute.

  All these values are returned in the form of ready-to-use lists.
- **get_calls_hour_perc:**
  This function takes as input values the users, the number of calls longer than one minute and the total number of calls.

  From the input data, this function calculates the calls of more than one minute per hour for each SDR (depending on the time of day, the programme takes into account whether there has been a break or not and divides by the time elapsed since the start of the working day), and the percentage of calls of each SDR that exceed one minute in duration.

  With this function and the previous one we can display all the data on the screen through the functions that print the corresponding metrics.
- **get_metrics:**
  This function takes as input values the users, the four filters specified in table 3[21] and the color of each profile.

  The function simultaneously executes the deals count in each filter we pass it to save time and resources, as this way the metrics load is not so tedious. Once it has taken all the counts, what the function does is to create a DataFrame with all the metrics (giving each SDR a row of the same).

[21] The filters are: won, lost, open and inactive business.

Once it has finished it calculates the conversion from the wins and losses of each SDR since we also want this metric and finally it adds the color assigned to each SDR to return the table with all the metrics.

- **get_points:**
  This function takes as input values the filter IDs of the earned filter and the users.

  The function goes through each filter analyzing the value of each business won because depending on the value the SDR gets a higher or lower amount of points[22], once it finds the value it assigns a score and stores them in a list.

  When all the values have been assigned to the list, they are added together to calculate the final points that each SDR has for the demos obtained.

  Finally, the function creates a DataFrame and stores all the data there for future use.

As can be seen, these functions only extract the data and clean it from the providers' API services but at no point do they display the information on the screen.

This task is taken care of by functions called "viewer functions", there are many of them scattered throughout the document and they all have the same structure, the following image will help us to understand it better::

```
# Function that prints conversion rates on the screen ----------------------------------------------------------------
def get_conv_test(u, df):
    metrics_df = df              # Rename the DataFrame
    cols = st.columns(len(u))    # Create the necessary columns

    for i, col in enumerate(cols):   # With this loop we set the information to each column
        with col:
            user = u[i]   # Set the user
            cerrados = metrics_df['Negocios perdidos'][i] + metrics_df['Demos Cerradas'][i] # Set the closed deals

            if cerrados == 0:   # if closed deals are 0 percentage is 0
                percentage = 0

            else:               # if closed deals are more than 0 we get the percentage from the DataFrame
                percentage = metrics_df['percentage'][i]

            st.metric(label=f'Conversión de {user}:',  # Display the information on the screen
                value='{}%'.format(percentage))
# --------------------------------------------------------------------------------------------------------------------
```

Image 10: Structure of a viewer function
Lluís Pellejà 2023

As we can see, the function needs as input parameters the users and the DataFrame where there is the information to show. The first action it performs is to create as many columns as there are users, so it will always adapt to any number of users (the app recommends not to exceed 10 per row).

[22] SDR points are calculated on the basis of the value their demos bring to the company. The criteria are as follows:

- if the fleet is 1-5, 0.3 points
- if the fleet is 6-10, 0.5 points
- if the fleet is 11-100, 1 point
- if the fleet is more than 100, 1.5 points.

From these points the sales incentive system is calculated for each profile.

Once the columns are created with a for loop filling in the information for each user, in this case before applying the information should be checked if the percentage is valid because if there are no businesses will give us a value Nan so it is checked first and if everything is OK the value is displayed on the screen.

Group SDR KPIs this month:
In the case of the group SDR KPIs this month module, the variables are more sparse as all SDRs are grouped together to see the total metrics for the month and draw some extra conclusions.

The variables that are needed are the following:

Variable	Usage
Origin	The origins from which deals come to Pipedrive
Created deals filter (current month)	The ID of the filter contains all the businesses created during the month.
Won filter ID by origin (current month)	The IDs of the CRM filters containing the business won by each origin
Lost filter ID by origin (current month)	The filter ID of the filter containing the missing by origin

Table 4: variables used in group SDR KPIs this month *module*
Lluís Pellejà 2023

In this case the app only executes two functions that serve to collect all the information of all the KPIs, these functions are:

- **get_metrics_group:**
  This function takes as input parameters all the variables in table 3, as it is a sparser module and does not need as much input data.

  The function works exactly like the *get_metrics* call of the previous module, it executes the deals counts simultaneously and creates a DataFrame to store the values in order to display them on the screen.

  In this case the function has to do an array due to API limitations[23] so there is a repeated source in the code.
- **get_lost_reasons_count:**
  This function takes as input value the IDs of the missing deal filters.

  With them, what it does is to go through the whole API response and store in a list all the reasons for lost deals. Once it has done all of them, what it does is to create a

[23] The CM API does not allow exporting more than 500 deals in the same api response, so for values exceeding these numbers we must do it in two parts and then join them in the DataFrame.

DataFrame of a single column and apply the function value_counts that allows us to know how many values of each one there are in the whole DataFrame in a quick way.

This method allows us to generate a graph to show the information on the screen.

In this module there are not the functions that exist in the previous module that show all the information on the screen, as in this module there is always the same number of columns and origins, so it is not necessary to change it every so often.

Group SDR KPIs past month:
In this module the application combines group metrics with individual metrics but from the previous month, this gives an overview of how the team is evolving month by month and allows the user to compare between the current month and the past month to see if there is improvement or not.

The variables required for this module are as follows:

Variable[24]	Usage
Users	This is the list of user names to be used in the frontend.
Colours	The color assigned to each user for the visualizations.
Won filter ID (last month)	The IDs of the CRM filters containing the business won by each SDR.
Open filter ID (last month)	The IDs of the CRM filters that contain the businesses that keep each SDR open.
Lost filter ID (last month)	The IDs of the CRM filters containing the business that each SDR has lost.
Imported deals (last month)	Total deals imported in the previous month
Lost reasons	All lost from the previous month without distinguishing by SDR

Table 5: variables used in Group SDR KPIs past month *module*
Lluís Pellejà 2023

To be able to do this module, there are three functions that the app executes in order to gather the necessary information and display it on the screen:

- **get_metrics_previous_month:**
  This function receives as input parameters all the variables in table 5 except the last two.

[24] These variables are the ones used in this part of the KPI module.

The operation of the function is exactly the same as the functions *get_metrics* and *get_metrics_group* of the previous modules.

This function collects all the information of the previous month in a table so that it can be put on the screen as correctly as possible.

- **get_points_previous_month:**
  This function receives as input parameters the users and the IDs of the last month's won filters.

  The function works in exactly the same way as the *get_points* functions in the previous module.

  This function collects all the information from the previous month in a table so that it can be put on the screen in the most correct way possible.
- **get_deal_count_last_month:**
  This function receives as input parameters the missing points of the last month.

  The function works in exactly the same way as the *get_lost_reasons_count* functions in the previous module.

  This function collects all the information from the previous month in a table so that it can be displayed as correctly as possible.

As can be seen, the module is very simple but necessary to see if the department is evolving or not and to be able to take important decisions in this respect.

The KPI - Support and Public Dashboard modules do not appear in this explanation, their functioning is exactly identical to the one explained in this section so there is no need to duplicate information, these modules were created to cover specific needs based on the request to Market Research, but the functioning is exactly the same.

Important SDR calls:
This is one of the most used and most recent modules of the app, it consists of a module where the most important calls of any SDR are stored at a simple click away.

These calls are used to make corrections to the call speeches of the profiles to improve their conversion rate, their phone safety and above all their communication skills.

This action is never stopped, that's why all profiles are available. The aim of this module is to make it easy for Market Research to access this type of information because if we have to find the calls by hand we waste an estimated 15 minutes a week, so in 15 seconds the calls are on screen with the name of the person who identifies them and the number to which the call is directed.

The first image that the app displays to the user when he enters the module is a short explanation and a row of buttons each with the name of a different SDR.

If the user presses any button the function called "*get_5_min_calls()*" is activated, this function does the following processes:

1. The Aircall identifier is assigned according to the button the user presses and the function is activated.
2. The function searches for all calls that belong to the user and have been made in the last 7 days.
3. The function collects three key pieces of information such as the call recording, the numbers and the person's name extracted from Pipedrive.
4. The function returns all values.

When the application has returned all the values, it groups the calls into four groups and adds the data it has collected on the screen.

The end user after about 15 seconds sees the following image:

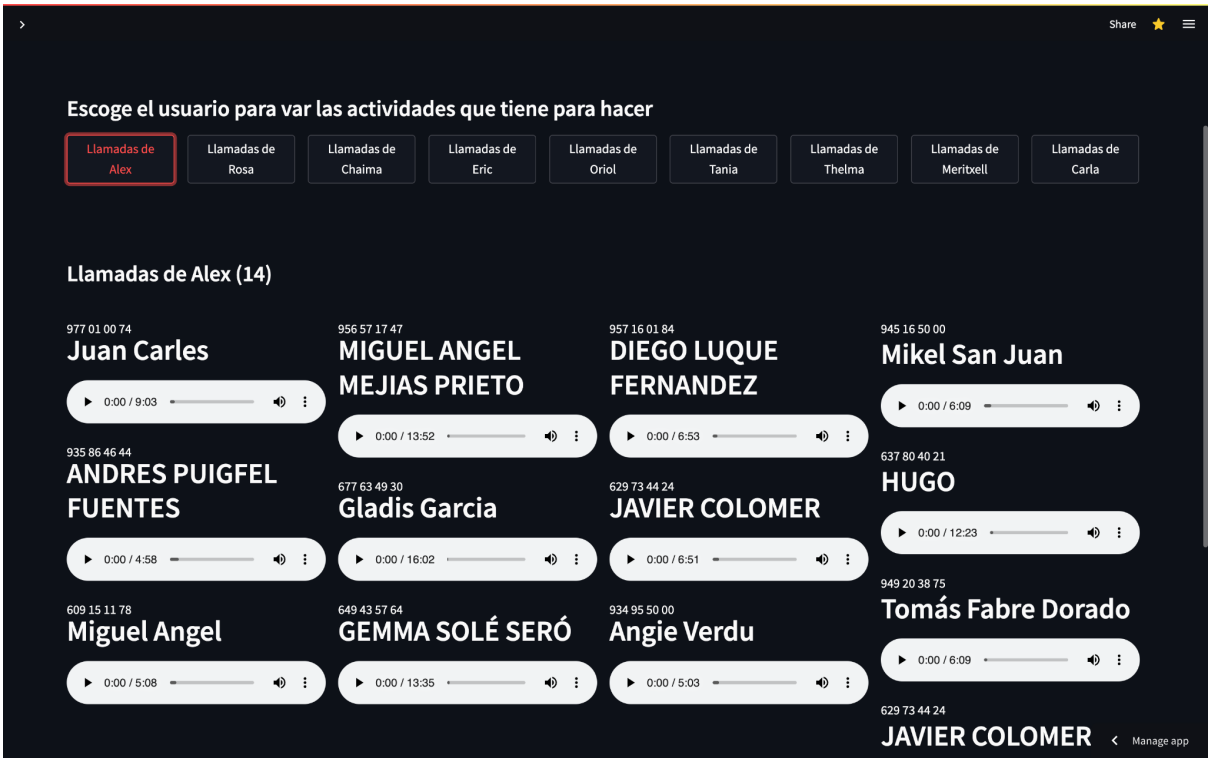


Image 11: Final user's view important calls module
Lluís Pellejà 2023

As can be seen in the image, the calls are grouped in four columns with the name and number of the person, in this case it looks like this because the capture has been made on a small screen.

Once loaded depending on the volume of data it will not load all the calls and will have to be restarted, this is because the free version of streamlit gives a maximum capacity and with this type of files it is easily exceeded.

6. CONCLUSION

This document reflects the creation of a very complete application for the needs for which it was created, as mentioned at the beginning of this document, the main objective is to optimize the time of the Market Research department staff.

This objective has been achieved as the application not only optimizes time in manual processes, but also optimizes time in each of the parts and processes of the department:

- Looking at the SDR KPIs
- Analyse calls
- Generate contacts
- Generate strategies
- Generate contact information

All Market Research processes have been improved thanks to the creation of this application. This is proven by the fact that the team has been using it daily for more than 3 months and already has a daily active user base of 4 users.

The application has fulfilled its objectives, and despite being a beta, it performs perfectly with what is asked of it.

The next steps for the application are to solve some connectivity issues and make it more accessible to the public that needs it, in order to attract new users so that they can also optimize their time.