Periple

Alexandre Morlat
dept. of Electrical Engineering
Hanyang University
Paris, France
alexandre.morlat@edu.devinci.fr

Henri Eloy
dept. of Electrical Engineering
Hanyang University
Versailles, France
henri.eloy@edu.devinci.fr

Aurélien Pouxviel
dept. of Electrical Engineering
Hanyang University
La Celle Saint Cloud, France
aurelien.pouxviel@edu.devinci.fr

Samuel Pariente
dept. of Electrical Engineering
Hanyang University
Levallois, France
samuel.pariente@edu.devinci.fr

Tristan D'antin
dept. of Electrical Engineering
Hanyang University
Paris, France
tristandantin@gmail.com

Abstract— This document is about our project "Periple". When you arrive in a new city and are looking for things to do or see, it's a pain to look around to find what to do. With periple, you will now be able to create your own tailored and intelligent trip that will only suggest things you will like!

Presentation video: https://youtu.be/MscPm15DSXk

Keywords—city, trip, intelligent

I. Introduction

A. Motivation

PERIPLE is a start-up idea that came to us when we were brainstorming on a topic for the AI and applications project. While we were exchanging on current innovative projects, we imagined a service to create a user-tailored tourism program. At the beginning of this semester in South Korea, such a tool would have proven to be greatly useful in order to optimize our tour days and to assist us in planning. Of course, effective tools are already established in the tourism market but we are thinking bigger. Moreover, we are convinced that artificial intelligence can revolutionize, or at least improve, the functioning of existing services.

B. Purpose

The idea is the following: to develop an artificial intelligence solution to predict what a tourist who wants to furnish his stay will like, and this from the analysis of a database, upstream.
The intelligent algorithm would feed on three factors to propose a suitable activity program:

- The general opinions already existing on the activities
- Its opinion on what it has liked or disliked
- The coherence of the proposed program (in terms of distance between each activity, price, etc.) But we are thinking bigger, and we even plan to develop a real powerful application, based on this famous algorithm, which would include the best of the market (visuals, functionalities, ergonomics) and possibly other essential functionalities (ticketing, GPS navigation, etc.).

Concretely, here is how we imagine a classic use of our application:

- The user launches the application and logs in/registers with his login and password.
- A quick questionnaire is launched on the kind of activities he is interested in (rather museums, monuments, nightlife, gourmet restaurant, fast food, etc.) and that he rates from 1 to 5 stars.
- Once the user's profile has been defined, he or she is offered a list of activities that he or she might enjoy.
- Once the user has gone to an activity, the application proposes to the user to click on Like/Dislike, which will further enhance the algorithm and propose more and more suitable activities. You will find through our blog the totality of our steps and the whole explicitation of various realizations from the obtaining and the modification of the database, the elaboration of the processing/prediction algorithm to the programming of a prototype interface. We wish you an excellent reading, up to our excitement for our project and we apologize in advance if some misunderstandings remain because of a sometimes-imperfect mastery of the English language.

II. Datasets

A. Introduction

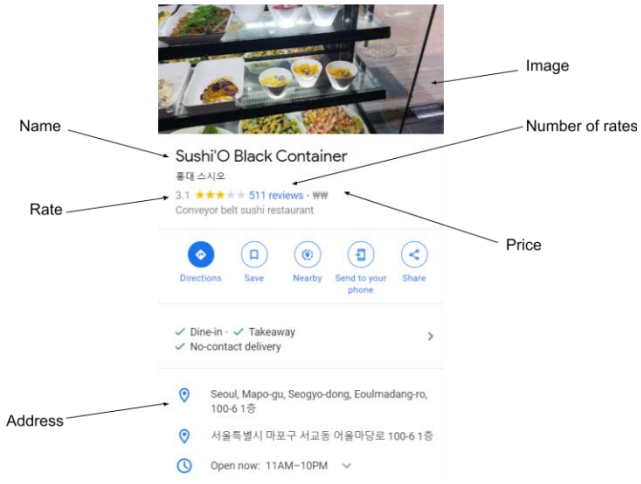
As we needed specific information which was not already on a dataframe, we had to build it. First of all we built a template dataframe looks like that:

	A	B	C	D	E	F	G	H	I
1	id	name	OriginalGrade	lagrade	MLGrade	gradeNbVoters	Price€	Location	numberofrates

J	K	L	M	N	O	P	Q	R
TimeH	Unmissable	PlaceID	Type 1	Type 2	Type 3	Type 4	photoID	image

B. *Google rating*

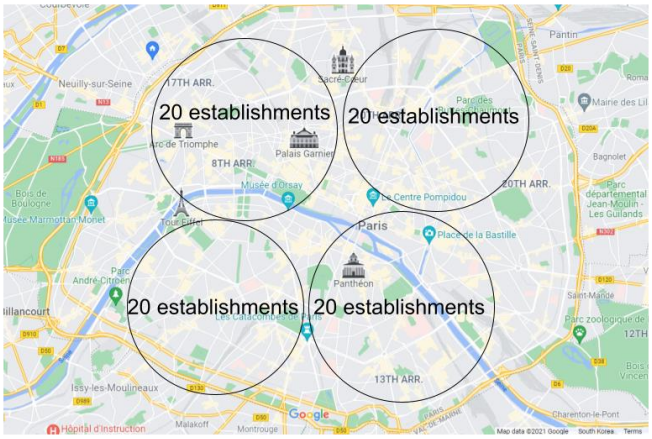
Our two most important variables are the name and de rating associated. We had two choices: google rating or trip advisor rating. As google API are more accessible, we choose Google:



How to take this information for a lot of establishments? As we saw in the introduction template, we need some types to define the places: catering gastronomic local; catering gastronomic global; catering casual local; catering casual global; catering fastfood; catering original; activity sport practice; activity sport watch; activity fun; activity relaxation; activity show cinema; activity show theater; nightlife club; nightlife bar; visit museum historical; visit museum modern ; visit monument exterior; visit monument interior; visit walk nature; visit walk urban.

C. *Google Place API*

Nearby fonction: a google api function which takes 20 locations near gps coordinate in a ray and related to Keywords. For each "types ligne" we will make a request. However, at the end 20 establishments for one line is not enough, so we split in 4 smaller rays:



D. *Code*

We took 80 establishments for 20 "type lines" and delete duplicates. Code here:

https://github.com/HenriEloy/PERIPLE/blob/8fc87fc18f8c55d24b6eea87df7c2bb683b16426/DataBases/builtdata.py

	A	B	C	D	E	F	G	H	I	J
1	id	name	OriginalGrade	lagrade	MLGrade	gradeNbVoter	Price€	Location	numberofrates	TimeH
2	0	Le Cinq	4.7	0	0	0	4	31 Av. Geo	1233	0
3	1	Restaurant	4.5	0	0	0	3	parc Bergs	181	0
4	2	Laurent	4.5	0	0	0	4	41 Av. Gab	369	0
5	3	Pierre Gag	4.5	0	0	0	4	6 Rue Balz	673	0
6	4	Le 39V	4.5	0	0	0	4	39 Av. Geo	334	0
7	5	L'Agapé	4.4	0	0	0	4	51 Rue Jou	316	0
8	6	Monsieur I	4.3	0	0	0	4	7 Rue de B	86	0
9	7	Restaurant	4.7	0	0	0		32 Av. Mat	99	0
10	8	Restaurant	4.6	0	0	0		18 Rue d'A	98	0
11	9	Palais Roya	4.6	0	0	0	4	110 Gal de	426	0
12	10	Apicius	4.4	0	0	0	4	20 Rue d'A	535	0

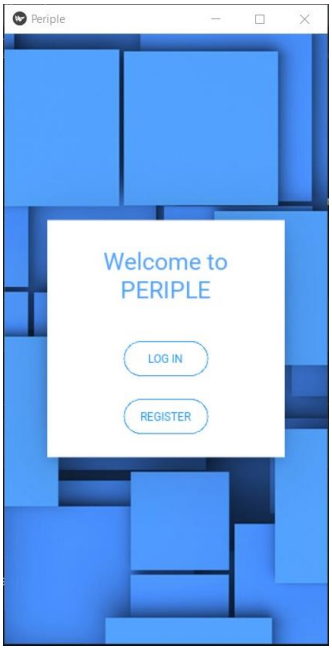
	K	L	M	N	O	P	Q	R	S	T
1	Unmissable	PlaceID	Type 1	Type 2	Type 3	Type 4	photoID	image		
2	0	ChIJqTQzIe	Catering	Gastronon	Gastronon	0	Aap_uECkJ	Le Cinq.png		
3	0	ChIJudQUF	Catering	Gastronon	Gastronon	0	Aap_uEAk'	Restaurant Le Gaigne.png		
4	0	ChIJlcyBCs	Catering	Gastronon	Gastronon	0	Aap_uEAP	Laurent.png		
5	0	ChIJn82-D	Catering	Gastronon	Gastronon	0	Aap_uEBXI	Pierre Gagnaire.png		
6	0	ChIJ-wegfs	Catering	Gastronon	Gastronon	0	Aap_uECN	Le 39V.png		
7	0	ChIJQ3-giL	Catering	Gastronon	Gastronon	0	Aap_uECo'	L'Agapé.png		
8	0	ChIJMfkUJ	Catering	Gastronon	Gastronon	0	Aap_uEB8	Monsieur Restaurant.png		
9	0	ChIJF81Lvc	Catering	Gastronon	Gastronon	0	Aap_uECJ(	Restaurant La Scène - Stéphanie		
10	0	ChIJnUjhH	Catering	Gastronon	Gastronon	0	Aap_uEDlZ	Restaurant Contraste.png		
11	0	ChIJ9Qib9	Catering	Gastronon	Gastronon	0	Aap_uED1'	Palais Royal Restaurant.png		
12	0	ChIJYaoSyf	Catering	Gastronon	Gastronon	0	Aap_uED2	Apicius.png		

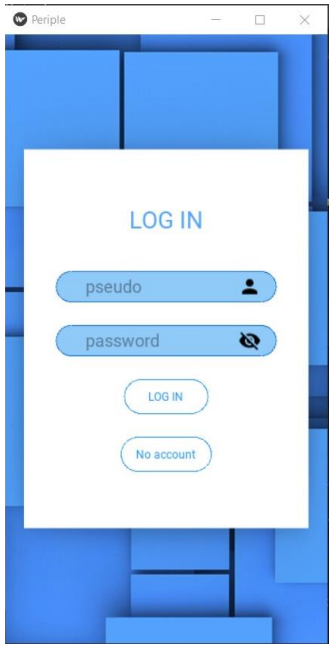
At the end we have 763 establishments of 20 different types associated with an image.

III. Methodology

To achieve this personal list of activities, we will use the data collected with google API (as we explained earlier), but we also obviously need data from the user. I will start by describing what a lambda user will see when opening our IA app, and then explain how we computed everything to find the perfect activities for him.

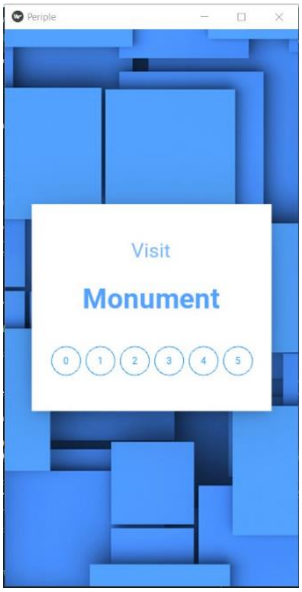
First, we ask the user to create an account because we want to save some data that he will give us and use it only for this person.



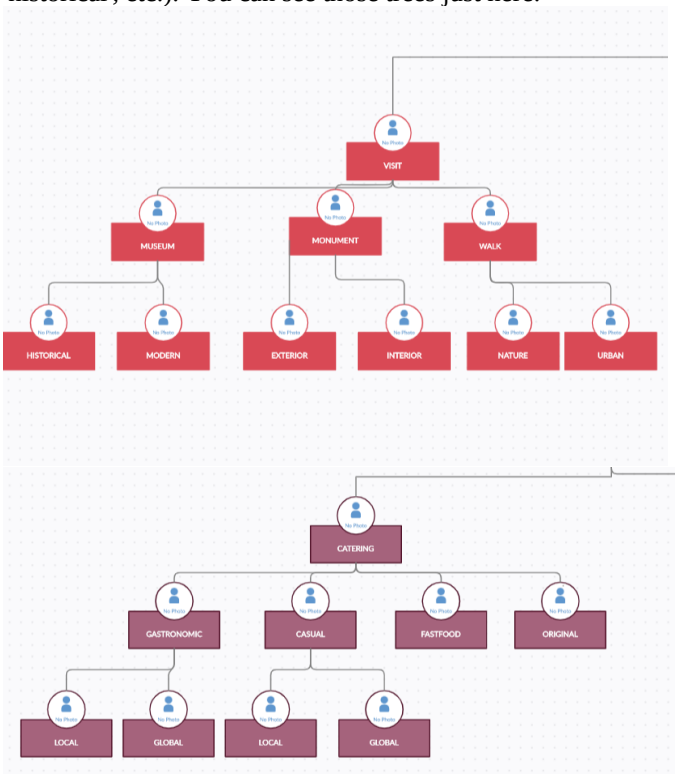


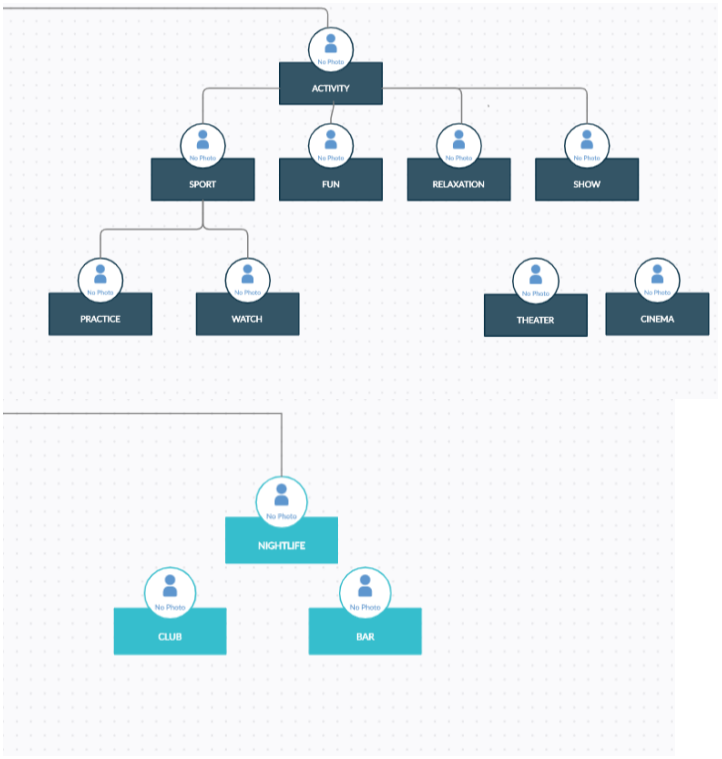
Identify applicable funding agency here. If none, delete this text box.

Once he is connected, we present him several 'types' of activities, and ask him to give a grade out of 5 for each of those types.



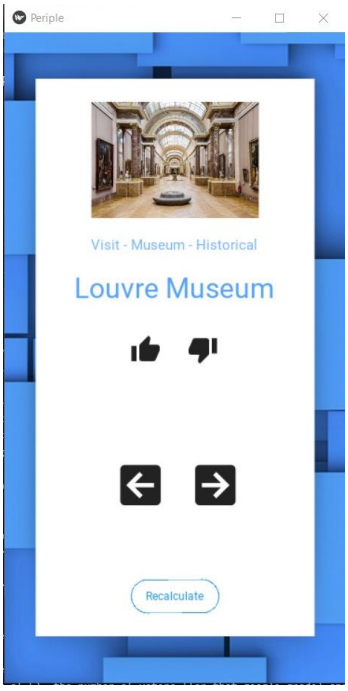
It's good to know that types are sorted and organized in a tree, starting with main types (such as 'Visit' or 'Catering') in which there are more focused types (such as 'museums', then 'historical', etc.). You can see those trees just here:





To simplify and save time for the user, we don't ask him to give a grade to a type if he gave a bad one to the 'parent' category.

When he has graded all the types, we will show him the first activity we think he will like and propose several options for each activity: to like or dislike the activity (after he did it or if he already did it on a previous trip), to see the next one, to come back to the last one, and finally, to recalculate everything, considering the activities he just liked or disliked.



That's it for our first version! Of course, we plan to go further after this semester, but we really focused on the algorithms and databases, to have a strong base for the future of PERIPLE.

Now, we must explain how we compute all our data, without using any of the known IA and machine learning models.

First, we have the database created with google API, we have 763 activities in Paris, and a lot of information about them. But for now, we just use several information: the name, the google grade (out of 5), the number of voters (for that google grade) and the three types for each activity Case in point: for the Louvre Museum, we get from google a grade of 4.7/5, with 223394 voters and the three types: Visit – Museum – Historical (We will also use the image of each activity for the display at the end, but it's useless for our algorithms) To store our data, we are computing an average grade, calculated with 5 grades (we can easily change the coefficients of each of those grades to get the mode accurate prediction possible).

A. Google grade

Not a lot of things to say about this one, it's just the grade given by hundreds of google users, we have it on the database built previously with google API. (Example: 4.7/5 for Louvre Museum) This grade has a coefficient of 0.8.

B. Voters grade

In order to separate the activities, and to favorize the most known (a 4.7 with 20 000 votes is more important than a 4.8 with 3 votes) For that, we used a logarithmic scale (between 0 and 10 voters, the grade is between 0 and 1, between 11 and 100 voters, the grade is between 1 and 2, between 101 and 1000 voters, the grade is between 2 and 3, etc.). Because very few activities have a lot of voters, and it was impossible to compute it normally (a few good and a lot of small grades). For example, with 223394 voters, Louvre Museum obviously gets a 5/5 This grade has a coefficient of 0.2

C. Machine learning grade

Do you remember when we propose to the user to like or dislike an activity he has done? When he does that, his opinion is used two times, and the first is right here. We save in a database all the opinions from the users, and we created a program considering all the binary decisions and computing a grade out of 5 for each activity. We compute using an easy method, every activity is noted in relation to the more and the less liked one. So, the one with the more likes gets a 5 and the one with the more dislikes gets a 0. And all other ones are between them. If this application were used by a lot of people, this machine learning could help us have our own grades for each activity, grades given by our users and not by all the users from internet. For now, we obviously don't have enough users to give this grade a real impact on the final grade, so the coefficient is for now 0.01.

D. Choices grade

As you can see, the three grades we already gave for each activity do not consider the choices or activities our logged user choose. They are the same for every user (the 3rd one can be changed by every user, for every user).

This 4th grade is based on the preferences the logged user gave us. He gave, for each type, a grade out of 5. We just have to make an average of each grade for each activity with those types! For example: if the user gave 3 to "Visit", 5 to "Museum" and 4 to "Historical", the Louvre Museum will get a "choice grade" of 4/5

That way, the choices of the user are considered. And the coefficient is 1 (the best one because we really want the user to visit things he really wants).

If it's the first time the program computes the perfect activities for a user, we stop here, and make an average of those 4 grades with associated coefficients. We can then easily sort by the final grade which is the best activity for our user!

E. The "Already done grade"

This last grade is very important. As we said, when the user like, or dislike an activity (which means he did it), a second thing appends: we will save the id of the activity, to be sure not to propose him this activity again, and we will improve (or lower) the 5th grade of all activities sharing one, two or three types with the activity already done! Of course, the more types there are in common, the more the grade will be changed!

We do that so when the user wants to visit again Paris (or any other city we will add after), or when he asks us to recalculate, we can consider what he already liked or disliked. It means, the more the user uses the application, the more it will be adapted to what he likes.

This grade has a coefficient of 0.4

We now have 5 different grades, and it's easy for us to know what the perfect activity for him will be!

F. Machine learning to compare

The method that we use is not the machine learning saw in class, so we try to use machine learning to compare with our results. Random forest First of all to use a random forest we had to convert our data to only numbers.
We made it thanks to this code :
https://github.com/HenriEloy/PERIPLE/blob/86d979d8285521e001fe033992f87054aa2ab5c2/Python%20Source%20Code/prediction%20data%20builder.py.
Secondly, we put randomly liked activities to the data frame. Then we train this data to make some prevision. At the end we put some activities the user liked; the algorithm predicts which activities the user will like after. The code: https://github.com/HenriEloy/PERIPLE/blob/4c3c8b02f1043e9858a6266e12213ca580d03b75/Python%20Source%20Code/random%20forest%20prediction.py

IV. Evaluation and Analysis

To analyse our Data base, we have use Python programming language (Jupyter Notebook). We used differents softwares like Numpy and Pandas for the data manipulation and result interpretation. The result we have found allowed us to have a better understanding of ou Data. Therefor those result will help us to creat an algorithm more accurate ans more precise.

Here you can find our study:

In [1]:
```
#DataBase Analysis
```

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime
from datetime import timedelta
```

In [3]:
```
#Visualisation of the data
with open('GDatabase.csv', newline='', encoding='UTF16') as csvfile:
    df = pd.read_csv(csvfile)
df
```

Out[3]:

	id	name	OriginalGrade	lagrade	MLGrade	Price€	Location	numberofrates	TimeH	Unmissable
0	0.0	Le Cinq	4.7	0	0	4.0	31 Av. George V, 75008 Paris, France	1233.0	0	0
1	1.0	Restaurant Le Gaigne	4.5	0	0	3.0	parc Bergson, 2 Rue de Vienne, 75008 Paris, Fr...	181.0	0	0
2	2.0	Laurent	4.5	0	0	4.0	41 Av. Gabriel, 75008 Paris, France	369.0	0	0
3	3.0	Pierre Gagnaire	4.5	0	0	4.0	6 Rue Balzac, 75008 Paris, France	673.0	0	0
4	4.0	Le 39V	4.5	0	0	4.0	39 Av. George V, 75008 Paris, France	334.0	0	0
...	...	...	...	...	...	...	...	...	...	...
759	759.0	Passage du Havre	4.1	0	0	0.0	109 Rue Saint-Lazare, 75009 Paris, France	8532.0	0	0
760	760.0	Expozoo, salon professionnel du marché de l'an...	5.0	0	0	0.0	2 Pl. de la Prte de Versailles, 75015 Paris, F...	1.0	0	0
761	761.0	Printemps Haussmann	4.2	0	0	0.0	64 Bd Haussmann, 75009 Paris, France	7546.0	0	0
762	762.0	Celio	4.2	0	0	0.0	CC Beaugrenelle, 19 Rue Linois, 75015 Paris, F...	120.0	0	0
763	763.0	Marché Poncelet	0.0	0	0	0.0	16 Rue Poncelet, 75017 Paris, France	0.0	0	0

764 rows × 17 columns

[4]:
```
#Firstly it could be Intersting to show to the user which Place is really unmissable in Paris
#For this we are going to attribute a logic output to the column unmissable
#This value is going to be determinated by the number of rates
#If there is a hight number of rate we can say that lot's of people have visited this place

df.loc[df[df['numberofrates']>50000].index,('Unmissable')]=1
df[df['numberofrates']>50000]['name']
```

[4]:
```
428    Luxembourg Gardens
638          Musée d'Orsay
639          Louvre Museum
680       Arc de Triomphe
681             Sacré-Cœur
682      Tuileries Garden
687           Eiffel Tower
714            La Villette
Name: name, dtype: object
```

```
#Now we can ask ourself if the unmissable places are the favorits places of the customers
#For this we are going to study the link between the grade of each place and if it's an unmissable place
#To study the correlation between a continue Variable (OriginalGrade) and a discrete Variable (Unmissable)
#We can use the Anova test.
import statsmodels.api
result = statsmodels.formula.api.ols('OriginalGrade ~ Unmissable',data =df[['OriginalGrade','Unmissable']]).fit
table = statsmodels.api.stats.anova_lm(result)
table
```

	df	sum_sq	mean_sq	F	PR(>F)
Unmissable	1.0	4.105734	4.105734	2.457246	0.117399
Residual	762.0	1273.201753	1.670868	NaN	NaN

```
#Here PR(>F) represent the p-value
#Usually if the p value is under 5% =0.05
#We can say that the variable are independent
#In this case the p value is equal 0.11
#It's mean that the grade doesn't have an influence on the fact that the place is Unmissable
#In other words the most famous place don't have the best grades
```

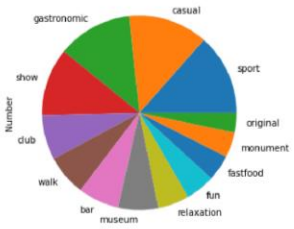
```
#With this data frame we can see what is the best categories
Ntype = 'Type 2'
Type1 = df[Ntype].value_counts().index
Average = np.zeros(np.shape(Type1)[0])
for i in range(np.shape(Type1)[0]):
    string = Type1[i]
    Average[i] = df[df[Ntype]==string]['OriginalGrade'].mean()

dfx = pd.DataFrame(columns=['Number','Average grade'],data = {'Number': df[Ntype].value_counts(), 'Average grad
dfx.sort_values(by ='Average grade',ascending=False)
#For example (Type 3) the people are interested about gastronomic local and they don't care about cinema and th
```

	Number	Average grade
monument	32	4.478125
gastronomic	95	4.476842
museum	51	4.347059
casual	101	4.322772
original	25	4.224000
fun	37	4.127027
walk	52	3.994231
relaxation	40	3.945000
sport	103	3.914563
fastfood	33	3.812121
bar	52	3.490385
show	86	3.181395
club	57	2.873684

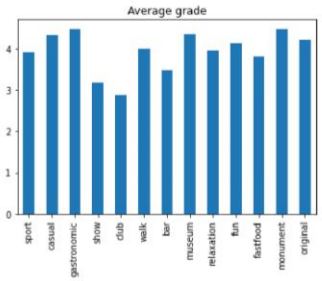
```
dfx['Number'].plot.pie(subplots=True,figsize=(5, 5))
#With this graph we can see that our data base contains different number of types
#FOr example we can present lot's of gastronomic but few fastfood restaurants.
```

```
array([<AxesSubplot:ylabel='Number'>], dtype=object)
```




```
dfx['Average grade'].plot.bar(subplots=True)
#With this bar graph we can see which type of places are best graded
```

```
array([<AxesSubplot:title={'center':'Average grade'}>], dtype=object)
```



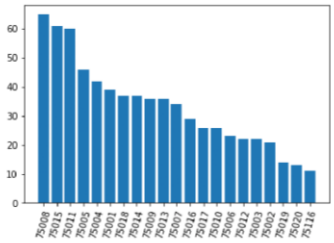
10]:
```
district= ['a']*df.shape[0]
for i in range(df.shape[0]):
    district[i] = df['Location'][i].split(',')[1].split(' ')[1]
    if  len(district[i]) != 5 :
        for word in df['Location'][i].split(' '):
            if word == 'Paris,':
                position = df['Location'][i].split(' ').index('Paris,')-1
                district[i] = df['Location'][i].split(' ')[position]
    if  len(district[i]) != 5:
         district[i]='NaN'



df_new_column = pd.DataFrame(district,columns=['district'])
dfT = df.join(df_new_column)

names, counts = zip(*dfT['district'].value_counts()[0:21].to_dict().items())
plt.bar(names, counts)
plt.xticks(rotation=75)
```

10]:
```
([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20],
 [Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, ''),
  Text(0, 0, '')])
```



11]:
```
#This bar Graph indicates that some districts in Paris are more touristics than others
#For example you can visit lot's of places in the 8th Arrondissement.
#If there is lot of thing to see in this district,
#our algorith will be more able to propose an hotel in this district for example
```

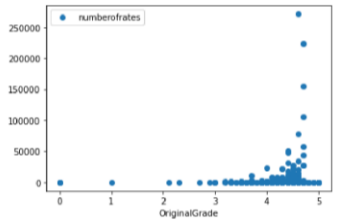
12]:
```
index_with_nan = dfT.index[df.isnull().any(axis=1)]
dfT.drop(index_with_nan,0, inplace=True)
dfT
```

12]:

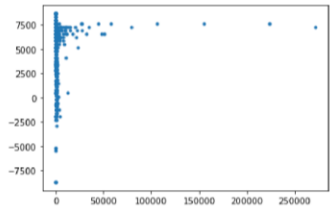
	id	name	OriginalGrade	lagrade	MLGrade	Price€	Location	numberofrates	TimeH	Unmissable
0	0.0	Le Cinq	4.7	0	0	4.0	31 Av. George V, 75008 Paris, France	1233.0	0	0
1	1.0	Restaurant Le Gaigne	4.5	0	0	3.0	parc Bergson, 2 Rue de Vienne, 75008 Paris, Fr...	181.0	0	0
2	2.0	Laurent	4.5	0	0	4.0	41 Av. Gabriel, 75008 Paris, France	369.0	0	0
3	3.0	Pierre Gagnaire	4.5	0	0	4.0	6 Rue Balzac, 75008 Paris, France	673.0	0	0
4	4.0	Le 39V	4.5	0	0	4.0	39 Av. George V, 75008 Paris, France	334.0	0	0
...	...	...	...	...	...	...	...	...	...	...
756	756.0	Peas and Love Toit de Yooma	4.3	0	0	0.0	51 Quai de Grenelle, 75015 Paris, France	8.0	0	0
757	757.0	Galeries Lafayette Champs-Élysées	4.3	0	0	0.0	60 Av. des Champs-Élysées, 75008 Paris, France	3216.0	0	0
759	759.0	Passage du Havre	4.1	0	0	0.0	109 Rue Saint-Lazare, 75009 Paris, France	8532.0	0	0
760	760.0	Expozoo, salon professionnel du marché de l'an...	5.0	0	0	0.0	2 Pl. de la Prte de Versailles, 75015 Paris, F...	1.0	0	0
761	761.0	Printemps Haussmann	4.2	0	0	0.0	64 Bd Haussmann, 75009 Paris, France	7546.0	0	0

406 rows × 18 columns

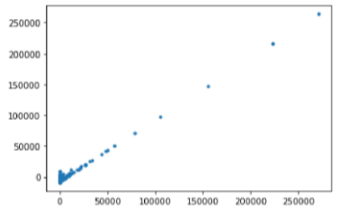
```
X = dfT['OriginalGrade']
y = dfT['numberofrates']
dfT.plot(x='OriginalGrade', y='numberofrates', style='o')
plt.savefig("pandas_scatter_plot_01.png", bbox_inches='tight', dpi=100)
```




```
import numpy as np
from sklearn.linear_model import LinearRegression
X = dfT[['Price€','OriginalGrade']]
y = dfT['numberofrates']
reg = LinearRegression().fit(X,y)
reg
plt.plot(y, reg.predict(X),'.')
plt.show()
#reg.score(X,y)
#reg.coef_
#reg.intercept_
#reg.predict(np.array([[3, 5]]))
#array([16.])
#Here we tryed to do a Linear regression but we can see that in the data just few places are rates more than 5
#It's why we can observe a distorted graph
#Prediction are not accurate on this case
```



```
plt.plot(y, y=reg.predict(X),'.')
plt.show()
```



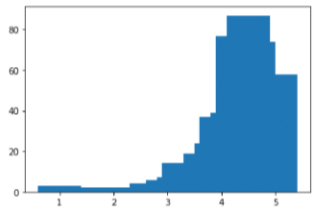
```
from scipy.stats import chi2_contingency
table = pd.crosstab(df["OriginalGrade"],df["Price€"])
result = chi2_contingency(table)
print("Statistic of the test : ",result[0])
print("p-value : ",result[1], ' = ', f"{result[1]:10.10f}")
print("degree of liberty : ",result[2])

#We can observe that the p-value is inferior to 5%
#So the price of the
#We can conclude that in general if you have to pay to see a place
#you can expect it to be better graded and so more interesting.
```

```
Statistic of the test :  229.14768565384506
p-value :  7.7939644626188448e-09  =  0.0000000078
degree of liberty :  120
```

```
#We don't use the grades equal to 0 because it's just mean ungraded
names, counts = zip(*df[df['OriginalGrade']!=0]['OriginalGrade'].value_counts().to_dict().items())
plt.bar(names, counts)
```

```
<BarContainer object of 30 artists>
```



```
Expectation=0
dictGrades = df[df['OriginalGrade']!=0]['OriginalGrade'].value_counts().to_dict()
names, counts = zip(*df[df['OriginalGrade']!=0]['OriginalGrade'].value_counts().to_dict().items())
for i in range(len(dictGrades)):
    Expectation = Expectation + (names[i]*counts[i])/sum(counts)

Expectation
print("The expectation of the grades is : ",Expectation)
```

```
The expectation of the grades is :  4.271509971509971
```

```
Variance = 0
for i in range(len(dictGrades)):
    Variance = Variance + (counts[i]/sum(counts))*(names[i]-Expectation)**2
print("The Variance of the grades is : ",Variance)
```

```
The expectation of the grades is :  0.33884643793475705
```

V. Related work

To achieve this project, we used several libraries:
Pandas to work on our data, using pandas dataframes, Kivy and KivyMD to create a very simple graphic interface. To develop the graphical interface, we first made graphical tests on the Figma tool (an interface design tool), then on the Bubble.io platform (no code tool making development). However, we were quickly limited to include all our algorithms and databases, that's why we continued Kivy. We also added a little library named "re" which allow us to verify if the email given by the user is a real one (when registering). All our data is stored in csv files locally created, but we started to use a server to host them online. But this is difficult to do it and modify our data afterwards, it's why we kept our data in local folders at least until we finish calibrating our machine learning.

VI. Conclusion / Discussion

To sum up this project, we have managed to operate in a structured way. We relied on tools such as communication tools, file sharing and task planning.
We divided the tasks between us, but all our work was linked, whether it is the algorithmic part, creation/finding of datasets, hosting thoses data, create a little graphic interface or analyse the data.

And as said before, we intend to go further. We really want to pursue this project with the idea of making a startup.

That's why we also worked on the development and not only the algorithmic part. We would be happy to have your feedback and we will continue to work hard on this project. When we will return to France, we intend to take the necessary steps to create the company and to get information from startup incubators, especially in our school which proposes some.