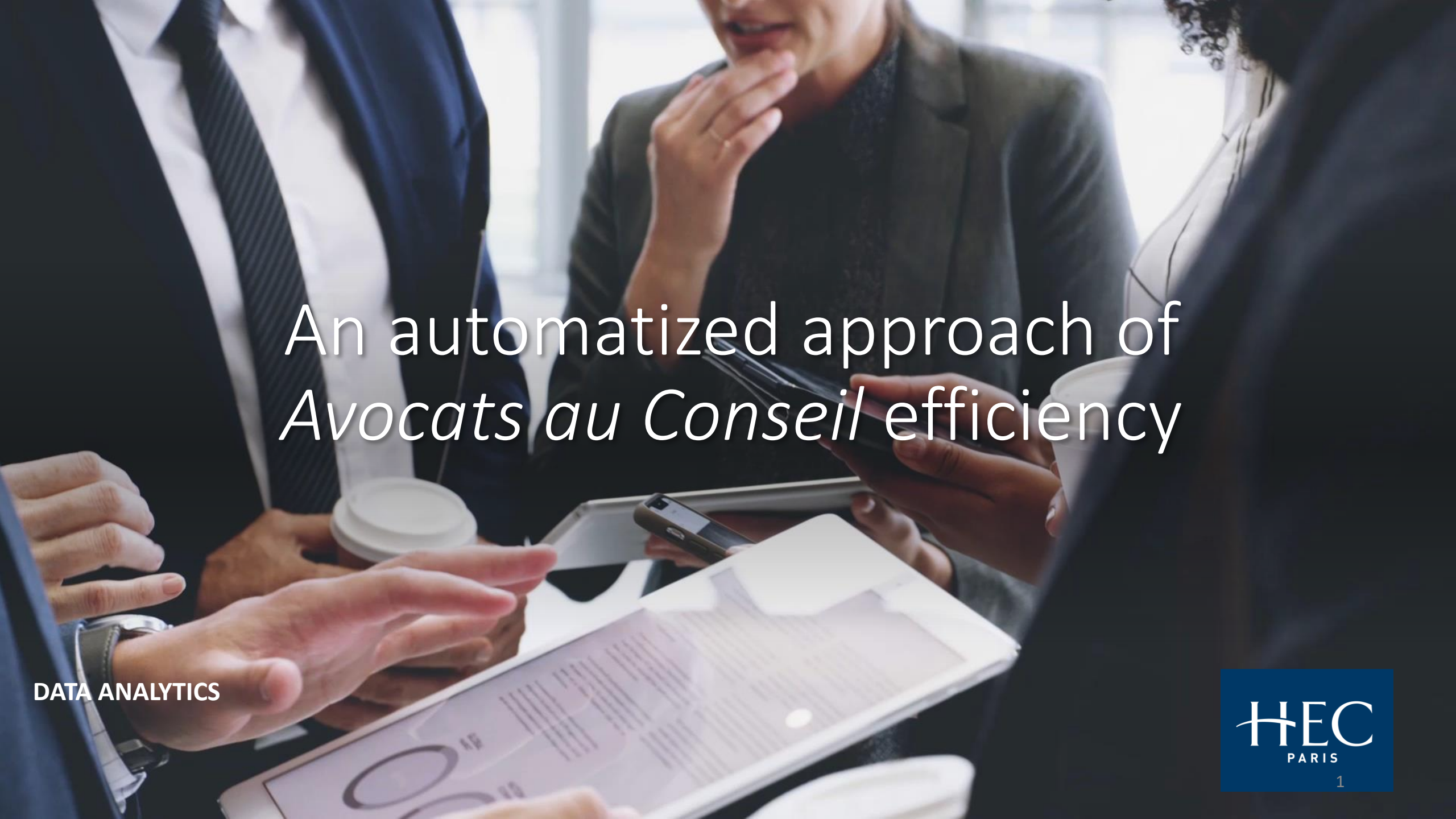


An automatized approach of *Avocats au Conseil* efficiency

DATA ANALYTICS

HEC PARIS

Context (1/2)

A very recent opening of court decisions in open data

Loi pour une République numérique,

Loi pour la programmation de la justice,

Décrets d'application avec 1ère échéance en 2021

A very technical litigation area

« Un contentieux en crise » (Réflexions sur la crise du contentieux économique, Benoît Remiche, 1984)

- Lack of technicals skills in parliaments
- « Inflation législative »

Monopoly of "Avocats au Conseil"

Context (2/2)

Légifrance

« *The data from Légifrance is made available for free re-use as laid down in the Order of 24 June 2014 on the free re-use of the legal databases of the Directorate for Legal and Administrative Information* »

2 472 293 court decisions

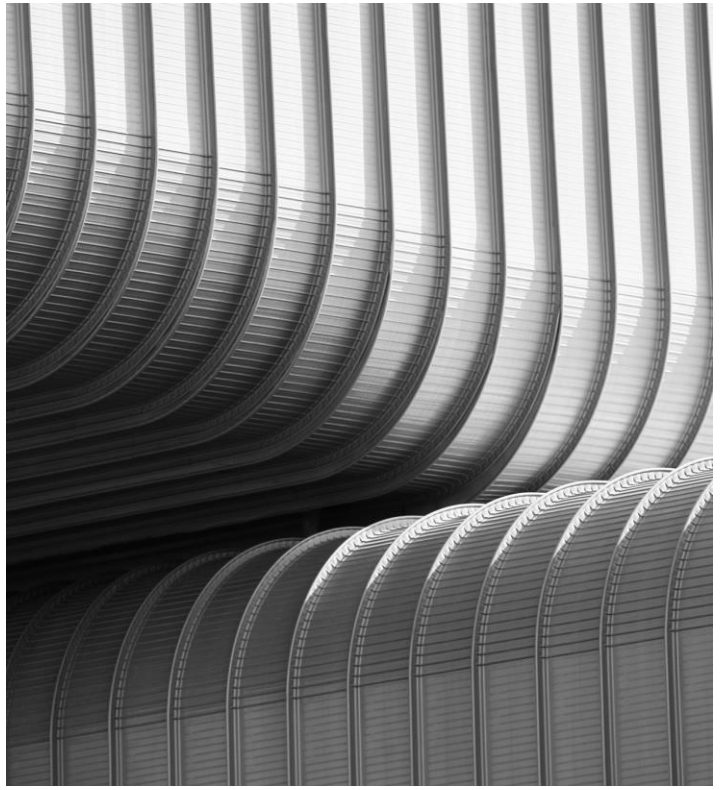
Our solution

Developping an automatical analysis of the Cassation decisions

Economical and commercial chamber

These 5 last years

The objective : assess the lawyer’s performance in a monopolistic market



1st step

Objective : scrapping all the decisions page

The criteria of the research are integrated manually in the Legifrance system, and the page is integrated in a soup

Therefore, creation of a list, and a loop will assess all the 55 pages of results

Integration of all the links of the decisions in a list

Law Data Analytics Project.py

```
# Law data analysis
# importing packages used (after installing)
import ...

# Opening the page
webpage = requests.get("https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&searchField=ALL&query=*&searchType=ALL&dateDecision=01%2F01%2F201
soup = BeautifulSoup(webpage.content)  # We create a soup element on that basis
driver = webdriver.Chrome(service=Service(ChromeDriverManager().install()))
# we extract the link of each case with a loop through the 54 pages
# we insert time sleep to prevent getting blocked
links = []
for i in range(1, 55):
    driver.get(
        "https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&searchField=ALL&query=*&searchType=ALL&dateDecision=01%2F01%2F2017+%3E+25%2F02%2
            i) + "&tab_selection=juri")

    soup = BeautifulSoup(driver.page_source)  # go to page number X of 54

    for link in soup.findAll('a', attrs={'href': re.compile("^/juri")}):  # loop to get all the links of that page
        links.append("https://www.legifrance.gouv.fr/" + link.get('href'))
# https://www.legifrance.gouv.fr/search/juri?tab_selection=juri&searchField=ALL&query=*&searchType=ALL&dateDecision=01%2F01%2F2017+%3E+25%2F02%2F2022&c
len(links) # number of decisions collected
# - The solution (rejection or acceptance or other)
# - The name of the law firms involved (both claimant and respondent)
# - Which law firm represents claimant and which represents respondent
# getting all the data
# first build an empty data frame
dataframe = pd.DataFrame()
```

2nd step

Prealable : creation of a dataframe who will integrate and process …

Decision number	Solution	Lawyer identification

Creation of a loop who will assess all the links listed and save all these informations listed

We create a soup, and use (i) html parser and, therefore, isolate the string to only have the core information

```
dataframe = pd.DataFrame()

# prepare the lists to be filled with the information of each decision
N_pourvoi = []
Solution = []
lawyers = []

# looping on all links, we get the informationn of each case and append the info to the lists defined above
# after that, we set the respective columns of the dataframe equal to the lists
for link in links:
    driver.get(str(link))
    soup = BeautifulSoup(driver.page_source,
                         'html.parser')
    info = soup.find("div", class_="frame-block print-sommaire")
    info = info.find_all_next(string=True, limit=11)
    lawyers.append(info[10])
    Solution.append(re.sub('[^a-zA-Z0-9 \n\.]', '', info[4].replace("Solution : ", "")))
    N_pourvoi.append(info[1].replace("N° de pourvoi : ", ""))

# Finalizing the dataframe
dataframe['n_pourvoi'] = N_pourvoi
dataframe['lawyers'] = lawyers
dataframe['solution'] = Solution

list_lawyers=[]
for i in range(dataframe.shape[0]):
    # Split by Law Firm and get the list of lawyers for each case
    list_lawyers.append([x.replace(",","").strip() for x in re.split('SAS|SCP|SARL|Me|SPFL',dataframe.loc[i,'lawyers'])])
```

3rd step

Objective : creation of the lawyers list

Success rate | Lawyer identification

We isolate all the core information;

we had to clear few times the information and create a « reduced » list

We had a new list with all the formally-cleared lawyers

```
# Finalizing the dataframe
dataframe['n_pourvoi'] = N_pourvoi
dataframe['lawyers'] = lawyers
dataframe['solution'] = Solution

list_lawyers=[]
for i in range(dataframe.shape[0]):
    # Split by Law Firm and get the list of lawyers for each case
    list_lawyers.append([x.replace(",","").strip() for x in re.split('SAS|SCP|SARL|Me|SPFL',dataframe.loc[i,'lawyers'])])

# remove "empty" lawyer
for i in list_lawyers:
    try:
        i.remove("")
    except ValueError:
        i=i

dataframe['lawyers_list']=list_lawyers

reduced = pd.DataFrame()
reduced_avocat = [] # will capture the name of the law firm
solution=[] # will save the solution as cassation or rejet

for i in range(dataframe.shape[0]):
    # Pick only cases in which there is maximum 2 law firms so to distinguish between claimant and defendant
    if len(dataframe['lawyers_list'][i]) == 2:
        reduced_avocat.append(dataframe['lawyers_list'][i])
```

4th step

Creation of a way to calculate the success rate

We process

“Cassation” as a **Claimant win**	“Rejet” as a **Defendant win**

And, therefore we have two lists with the number of wins per lawyer

```
solution=[] # will save the solution as cassation or rejet

for i in range(dataframe.shape[0]):
    # Pick only cases in which there is maximum 2 law firms so to distinguish between claimant and defendant
    if len(dataframe['lawyers_list'][i]) == 2:
        reduced_avocat.append(dataframe['lawyers_list'][i])
        if re.search("Cassation", dataframe['solution'][i])!=None:
            solution.append("Cassation")
        elif re.search("Rejet", dataframe['solution'][i])!=None:
            solution.append("Rejet")
        else:
            solution.append("Other")

reduced['lawyers']=reduced_avocat
reduced['solution']=solution

# classify the lawyer as claimant or defendant based on the position in the list
claim=[]
defendant=[]

for avocats in reduced['lawyers']:
    claim.append(avocats[0])
    defendant.append(avocats[1])

reduced['claim']=claim
reduced['defendant']=defendant
```

5th step

Extract both tables

Merging

Export to excel

BENALOUANE - FREY - MAURER - NICOLAS - VALENCE Data analytics project.py

```
reduced['claim']=claim
reduced['defendant']=defendant

# extract two separate table for claimant and for defendant, grouping by claimant or defendant and solution (counting)
table_defendant=reduced.groupby(["defendant","solution"]).count().drop(columns="claim").reset_index().rename(columns={"lawyers":"cases","defendant":"la
table_claim=reduced.groupby(["claim","solution"]).count().drop(columns="defendant").reset_index().rename(columns={"lawyers":"cases","claim":"lawyer"})

# count the wins for each lawyer in all the cases in which he was a claimant (wins if solution is cassation)
wins=[]
lawyer=[]
for i in range(table_claim.shape[0]):
    if table_claim['solution'][i]=="Cassation":
        wins.append(table_claim['cases'][i])
        lawyer.append(table_claim['lawyer'][i])
claimant_cassation=pd.DataFrame({"lawyer":lawyer, "wins":wins})

wins=[]
lawyer=[]
for i in range(table_defendant.shape[0]):
    if table_defendant['solution'][i]=="Rejet":
        wins.append(table_defendant['cases'][i])
        lawyer.append(table_defendant['lawyer'][i])
defendant_rejet=pd.DataFrame({"lawyer":lawyer, "wins":wins})

table = defendant_rejet.merge(claimant_cassation, on="lawyer", how="outer").fillna(0)
table['wins']=table["wins_x"]+table["wins_y"]
table=table.drop(columns=["wins_x","wins_y"]).groupby("lawyer").sum().reset_index().sort_values("wins", ascending=False).loc[table['lawyer']!="0"]
tot_cases=(reduced.groupby(['claim']).count().drop(columns=["defendant","solution"]).reset_index().rename(columns={"lawyers":"cases","claim":"lawyer"})
tot_cases['total']=tot_cases['cases_x']+tot_cases['cases_y']
```

BENALOUANE - FREY - MAURER - NICOLAS - VALENCE Data analytics project.py

```
lawyer=[]
for i in range(table_defendant.shape[0]):
    if table_defendant['solution'][i]=="Rejet":
        wins.append(table_defendant['cases'][i])
        lawyer.append(table_defendant['lawyer'][i])
defendant_rejet=pd.DataFrame({"lawyer":lawyer, "wins":wins})

table = defendant_rejet.merge(claimant_cassation, on="lawyer", how="outer").fillna(0)
table['wins']=table["wins_x"]+table["wins_y"]
table=table.drop(columns=["wins_x","wins_y"]).groupby("lawyer").sum().reset_index().sort_values("wins", ascending=False).loc[table['lawyer']!="0"]
tot_cases=(reduced.groupby(['claim']).count().drop(columns=["defendant","solution"]).reset_index().rename(columns={"lawyers":"cases","claim":"lawyer"})
tot_cases['total']=tot_cases['cases_x']+tot_cases['cases_y']
table=table.merge(tot_cases.drop(columns=['cases_x','cases_y']),how="left",on="lawyer")
table['proportion']=table['wins']/table['total']*100
table['proportion']=table['proportion'].map('{:,.2f}%'.format)

# import xlsxwriter module


# Creating the xlsx
workbook = xlsxwriter.Workbook('AvocatsSuccess.xlsx')
table.to_excel('AvocatsSuccess.xlsx')
```

Technical issues

Perspectives ?

MELISSA-SANDRA BENALOUANE, MICHEL FREY,
ALEXANDRE MAURER, EMILE NICOLAS,
DYLAN VALENCE

HEC PARIS