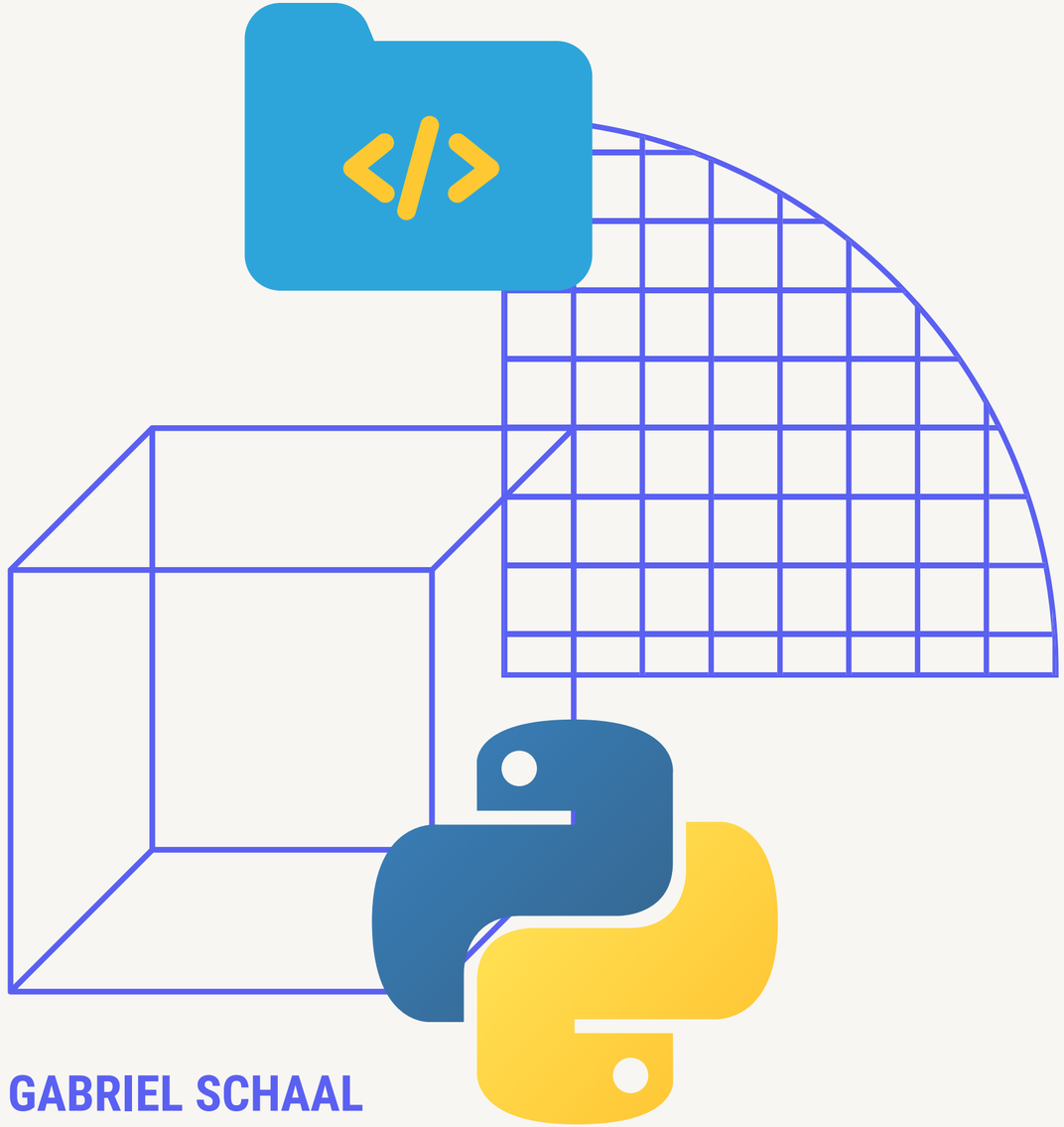
LEGAL DATA ANALYSIS
PR. DAMIEN CHARLOTIN

APRIL 5, 2024

PENALTIES FOR TAX ABUSE

Abuse of rights under article L. 64 of the LPF & sanctions under article 1729 of the CGI



GABRIEL SCHAAL
DEBORAH ZBILI

NOTION & RESEARCH OBJECTIVE

01.

METHODOLOGY

02.

ANALYSIS

03.

CONCLUSION

04.

TAX ABUSE : A general Framework

DEFINITION

Deviation under Article L. 64 of the LPF, requiring two cumulative criteria:

- For Fictitiousness
- For Fraud Against the Law

PENALTIES

- Restoration and Taxation
- **40% or 80% ?**
  - General Penalty: A non-deductible penalty equal to 80% of the evaded tax amount
  - Reduced Penalty for "Passive" Taxpayers

KEY WORDS

“manquement délibéré”

“manoeuvres frauduleuses”

Research objective

Initial objective : ***Investigate the proportion of application of the 40% or 80% penalty in tax litigation via a scrap from Ariane Web***

Issue : penalties are grouped together under the same article and key words are inexploitable

Follow up: How often are motions to contest these 40% or 80% penalties rejected or accepted by the Court ?

- Scrapping using Selenium ?
- Front end / back end use

METHODOLOGY - Scrapping

Two attempts at **scrapping**, is **Selenium** relevant in our search ?

How it works	
Selenium, a Python Library	Automates web browser control, opening sessions in the background to navigate websites as a human user
Process	Direct the browser to the Conseil d'Etat website, search for terms like "abus de droit fiscal" (abuse of tax law), and interact with elements such as buttons and links to collect data.

Limitations

Pagination:
Must manually navigate through multiple pages of results, a challenge if the number of pages changes.

Maintenance:
Less flexible and requires updates to the script if the website's structure changes.

A glance at our code lines using Selenium

```
[ ]  # Initialize an empty list to hold all scraped data
     all_data = []

     # Function to extract and return data from a row
     def extract_data_from_row(row):
         # Extract all 'td' elements that are not headers
         cells = row.find_all('td')
         # Extract text from each cell
         extracted_data = [cell.get_text(strip=True) for cell in cells]
         return extracted_data

     # Main scraping loop
     try:
         while True:
             # Wait for the dynamic content to load
             WebDriverWait(driver, 10).until(
                 EC.presence_of_element_located((By.CSS_SELECTOR, "tr[ng-repeat-start]"))
             )

             # Now use BeautifulSoup to parse the page source
             soup = BeautifulSoup(driver.page_source, 'html.parser')

             # Find all rows of interest
             rows = soup.select('tr[ng-repeat-start]')

             # Iterate over each row and extract data
             for row in rows:
                 row_data = extract_data_from_row(row)
                 all_data.append(row_data)

             # Attempt to find and click the 'Next' button, if it exists
             next_button = driver.find_elements(By.XPATH, "//button[contains(text(), 'Next') or @aria-label='Next']")
             if next_button:
```

A glance at our code lines using Selenium

```
            row_data = extract_data_from_row(row)
            all_data.append(row_data)

        # Attempt to find and click the 'Next' button, if it exists
        next_button = driver.find_elements(By.XPATH, "//button[contains(text(), 'Next') or @aria-label='Next']")
        if next_button:
            next_button[0].click()
        else:
            break  # Exit loop if there's no 'Next' button

finally:
    driver.quit()  # Ensure the driver is quit at the end

# Convert the list of data into a pandas DataFrame
df = pd.DataFrame(all_data, columns=['Rank', 'DecisionType', 'Section', 'Role', 'Date', 'CaseNumber', 'Category'])  # Adjust columns as needed
print(df.head())

# Here you may save the dataframe to a file or perform other operations as required.
```

```
  Rank DecisionType         Section              Role        Date  \
0    1       Arrêts       CAA DOUAI           Chambre  05/03/2024
1    2    Décisions  Conseil d État          3-8  CHR  26/02/2024
2    3    Décisions  Conseil d État          3-8  CHR  15/02/2024
3    4    Décisions  Conseil d État  Juge des référés  14/02/2024
4    5    Décisions  Conseil d État         10-9  CHR  09/02/2024

    CaseNumber Category
0    22DA01934       C+
1       469858        B
2       454475        B
3       491005        C
4  472346(...)        C
```

```
4  472346(...)        C
```

```
df
```

	Rank	DecisionType	Section	Role	Date	CaseNumber	Category
0	1	Arrêts	CAA DOUAI	Chambre	05/03/2024	22DA01934	C+
1	2	Décisions	Conseil d État	3-8 CHR	26/02/2024	469858	B
2	3	Décisions	Conseil d État	3-8 CHR	15/02/2024	454475	B
3	4	Décisions	Conseil d État	Juge des référés	14/02/2024	491005	C
4	5	Décisions	Conseil d État	10-9 CHR	09/02/2024	472346(...)	C
5	6	Décisions	Conseil d État	Juge des référés	10/01/2024	490477	C
6	7	Décisions	Conseil d État	Juge des référés	21/12/2023	489990	C
7	8	Décisions	Conseil d État	5-6 CHR	21/12/2023	473466	C
8	9	Décisions	Conseil d État	5ème CHS	21/12/2023	470132	C
9	10	Décisions	Conseil d État	6-5 CHR	18/12/2023	451947	C

METHODOLOGY - Scrapping

A focus on front end & back end differences : **Bypassing Ariane’s user interface**

Overview	
Backend vs. Frontend	The backend manages the application's server-side, including logic, data storage, and system interactions, while the frontend is the user interface for direct interaction.
Direct Backend Access	By accessing the backend directly rather than the frontend, we aim to communicate directly with the server hosting the Conseil d'État's data. This approach enables retrieval of more complete and structured information, free from the user interface's constraints.

METHODOLOGY - Scrapping

Bypassing Ariane’s user interface : *Key Advantage*

REFUSAL DECISIONS IDENTIFICATION

A decision's status as a refusal (décision de rejet) is identifiable directly in the backend, not visible in the frontend. This critical distinction is only observable in the legal basis or enactment (dispositif) of the decision.

WHY SELENIUM IS NOT SUITABLE

Limitation in Detecting Refusals:
Selenium's interaction with the frontend is inadequate for our purpose because a refusal can occur anywhere within the decision document, making it impractical for systematically identifying refusal decisions.

METHODOLOGY - `requests`


```python
import requests

# URL de l'API ou du service web à interroger
url = "https://www.conseil-etat.fr/xsearch?"

# Paramètres pour la chaîne de requête de la requête POST
params = {
    "advanced": "1",
    "type": "json",
    "SourceStr4": "AW_DCA",
    "synonyms": "true",
    "scmode": "smart",
    "SkipCount": "50",
    "SkipFrom": "0",
    "sort": "SourceDateTime1.desc,SourceStr5.desc",
    "add.text": "1729 du code général des impôts déchargé"
}

# Envoi de la requête POST et réception de la réponse
response = requests.post(url, params=params)

# Initialisation des compteurs pour "Rejet" et le total des décisions
count_rejet = 0
total_decisions = 0

# Vérification du statut de la réponse
if response.status_code == 200:
    # Transformation de la réponse en JSON
```

... Initialisation de Backend Google Compute

“request” use : By sending a request directly to the backend(requests.post(url, params=params)), data can be accessed more efficiently than by automated navigation via the frontend. This avoids the loading of unnecessary resources such as images and layout scripts.

Data analysis

```
imp  # requests

# URL de l'API ou du service web à interroger
url = "https://www.conseil-etat.fr/xsearch?"

# Paramètres pour la chaîne de requête de la requête POST
params = {
 "advanced": "1",
 "type": "json",
 "SourceStr4": "AW_DCA",
 "synonyms": "true",
 "scmode": "smart",
 "SkipCount": "50",
 "SkipFrom": "0",
 "sort": "SourceDateTime1.desc,SourceStr5.desc",
 "add.text": "1729 du code général des impôts déchargé"
}

# Envoi de la requête POST et réception de la réponse
response = requests.post(url, params=params)

# Initialisation des compteurs pour "Rejet" et le total des décisions
count_rejet = 0
total_decisions = 0

# Vérification du statut de la réponse
if response.status_code == 200:
 # Transformation de la réponse en JSON
 data = response.json()

 # Vérification que le champ "Documents" est présent dans les données
 if "Documents" in data:
 # Parcours de chaque document retourné par la requête
 for document in data["Documents"]:
 # Incrément du compteur du total des décisions
 total_decisions += 1

 # Vérification si "SourceStr12" est présent et équivaut à "Rejet"
 if document.get("SourceStr12") == "Rejet":
 count_rejet += 1

 # Affichage du nombre de "Rejet" et du total des décisions
 print(f"Nombre de 'Rejet' : {count_rejet}")
 print(f"Total des décisions : {total_decisions}")
else:
 print(f"Erreur lors de la requête : {response.status_code}")
```

```
    data = response.json()

    # Vérification que le champ "Documents" est présent dans les données
    if "Documents" in data:
        # Parcours de chaque document retourné par la requête
        for document in data["Documents"]:
            # Incrément du compteur du total des décisions
            total_decisions += 1

            # Vérification si "SourceStr12" est présent et équivaut à "Rejet"
            if document.get("SourceStr12") == "Rejet":
                count_rejet += 1

    # Affichage du nombre de "Rejet" et du total des décisions
    print(f"Nombre de 'Rejet' : {count_rejet}")
    print(f"Total des décisions : {total_decisions}")
else:
    print(f"Erreur lors de la requête : {response.status_code}")
```

```
Nombre de 'Rejet' : 24
```

19%

Reject Rate

Based on a total of 134 Court of Appeal decisions referenced on the Ariane Web site

CONCLUSION

Interpreting figures : Ariane web scrapping shows that only 19% of requests to contest a penalty for abuse of tax law are rejected. We can therefore conclude that the courts are favorable to this type of request. However, these figures must be qualified. They do not distinguish between requests to cancel all penalties and requests to reduce penalties from 80% to 40%.

“request” use : This code model can be reused in the future. Ariane web is one of the few databases to open up access to the back end. It may therefore be appropriate to re-use this code in several months or years' time, in order to monitor the tendency of courts to uphold a claimant's right to challenge a penalty for abuse of tax law.