BEE2041 Empirical Project Blog

In [255]:
```
pip install econml selenium --upgrade
```

```
Collecting econml
  Downloading econml-0.15.0-cp38-cp38-win_amd64.whl (2.0 MB)
Requirement already up-to-date: selenium in c:\users\socor\anaconda3\lib\site-packages (4.19.0)
Requirement already satisfied, skipping upgrade: numpy in c:\users\socor\anaconda3\lib\site-packag
es (from econml) (1.18.5)
Requirement already satisfied, skipping upgrade: scipy>1.4.0 in c:\users\socor\anaconda3\lib\site-
packages (from econml) (1.5.0)
Requirement already satisfied, skipping upgrade: joblib>=0.13.0 in c:\users\socor\anaconda3\lib\si
te-packages (from econml) (0.16.0)
Collecting sparse
  Downloading sparse-0.15.1-py2.py3-none-any.whl (116 kB)
Requirement already satisfied, skipping upgrade: pandas>1.0 in c:\users\socor\anaconda3\lib\site-p
ackages (from econml) (1.0.5)
Note: you may need to restart the kernel to use updated packages.

ERROR: scikit-learn 1.3.2 has requirement joblib>=1.1.1, but you'll have joblib 0.16.0 which is in
compatible.
```

```
Collecting shap<0.44.0,>=0.38.1
  Downloading shap-0.43.0-cp38-cp38-win_amd64.whl (447 kB)
Requirement already satisfied, skipping upgrade: statsmodels>=0.10 in c:\users\socor\anaconda3\lib
\site-packages (from econml) (0.11.1)
Collecting scikit-learn<1.5,>=1.0
  Downloading scikit_learn-1.3.2-cp38-cp38-win_amd64.whl (9.3 MB)
Collecting lightgbm
  Downloading lightgbm-4.3.0-py3-none-win_amd64.whl (1.3 MB)
Requirement already satisfied, skipping upgrade: urllib3[socks]<3,>=1.26 in c:\users\socor\anacond
a3\lib\site-packages (from selenium) (2.2.1)
Requirement already satisfied, skipping upgrade: typing_extensions>=4.9.0 in c:\users\socor\anacon
da3\lib\site-packages (from selenium) (4.10.0)
Requirement already satisfied, skipping upgrade: trio-websocket~=0.9 in c:\users\socor\anaconda3\l
ib\site-packages (from selenium) (0.11.1)
Requirement already satisfied, skipping upgrade: certifi>=2021.10.8 in c:\users\socor\anaconda3\li
b\site-packages (from selenium) (2024.2.2)
Requirement already satisfied, skipping upgrade: trio~=0.17 in c:\users\socor\anaconda3\lib\site-p
ackages (from selenium) (0.25.0)
Requirement already satisfied, skipping upgrade: numba>=0.49 in c:\users\socor\anaconda3\lib\site-
packages (from sparse->econml) (0.50.1)
Requirement already satisfied, skipping upgrade: python-dateutil>=2.6.1 in c:\users\socor\anaconda
3\lib\site-packages (from pandas>1.0->econml) (2.8.1)
Requirement already satisfied, skipping upgrade: pytz>=2017.2 in c:\users\socor\anaconda3\lib\site
-packages (from pandas>1.0->econml) (2020.1)
Requirement already satisfied, skipping upgrade: tqdm>=4.27.0 in c:\users\socor\anaconda3\lib\site
-packages (from shap<0.44.0,>=0.38.1->econml) (4.47.0)
Collecting slicer==0.0.7
  Downloading slicer-0.0.7-py3-none-any.whl (14 kB)
Requirement already satisfied, skipping upgrade: cloudpickle in c:\users\socor\anaconda3\lib\site-
packages (from shap<0.44.0,>=0.38.1->econml) (1.5.0)
Collecting packaging>20.9
  Downloading packaging-24.0-py3-none-any.whl (53 kB)
Requirement already satisfied, skipping upgrade: patsy>=0.5 in c:\users\socor\anaconda3\lib\site-p
ackages (from statsmodels>=0.10->econml) (0.5.1)
Requirement already satisfied, skipping upgrade: threadpoolctl>=2.0.0 in c:\users\socor\anaconda3
\lib\site-packages (from scikit-learn<1.5,>=1.0->econml) (2.1.0)
Requirement already satisfied, skipping upgrade: pysocks!=1.5.7,<2.0,>=1.5.6; extra == "socks" in
c:\users\socor\anaconda3\lib\site-packages (from urllib3[socks]<3,>=1.26->selenium) (1.7.1)
Requirement already satisfied, skipping upgrade: exceptiongroup; python_version < "3.11" in c:\use
rs\socor\anaconda3\lib\site-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied, skipping upgrade: wsproto>=0.14 in c:\users\socor\anaconda3\lib\sit
e-packages (from trio-websocket~=0.9->selenium) (1.2.0)
Requirement already satisfied, skipping upgrade: attrs>=23.2.0 in c:\users\socor\anaconda3\lib\sit
e-packages (from trio~=0.17->selenium) (23.2.0)
Requirement already satisfied, skipping upgrade: sniffio>=1.3.0 in c:\users\socor\anaconda3\lib\si
te-packages (from trio~=0.17->selenium) (1.3.1)
Requirement already satisfied, skipping upgrade: cffi>=1.14; os_name == "nt" and implementation_na
me != "pypy" in c:\users\socor\anaconda3\lib\site-packages (from trio~=0.17->selenium) (1.14.0)
Requirement already satisfied, skipping upgrade: sortedcontainers in c:\users\socor\anaconda3\lib
\site-packages (from trio~=0.17->selenium) (2.2.2)
Requirement already satisfied, skipping upgrade: outcome in c:\users\socor\anaconda3\lib\site-pack
ages (from trio~=0.17->selenium) (1.3.0.post0)
Requirement already satisfied, skipping upgrade: idna in c:\users\socor\anaconda3\lib\site-package
s (from trio~=0.17->selenium) (2.10)
Requirement already satisfied, skipping upgrade: llvmlite<0.34,>=0.33.0.dev0 in c:\users\socor\ana
conda3\lib\site-packages (from numba>=0.49->sparse->econml) (0.33.0+1.g022ab0f)
Requirement already satisfied, skipping upgrade: setuptools in c:\users\socor\anaconda3\lib\site-p
ackages (from numba>=0.49->sparse->econml) (49.2.0.post20200714)
Requirement already satisfied, skipping upgrade: six>=1.5 in c:\users\socor\appdata\roaming\python
\python38\site-packages (from python-dateutil>=2.6.1->pandas>1.0->econml) (1.15.0)
Requirement already satisfied, skipping upgrade: h11<1,>=0.9.0 in c:\users\socor\anaconda3\lib\sit
e-packages (from wsproto>=0.14->trio-websocket~=0.9->selenium) (0.14.0)
Requirement already satisfied, skipping upgrade: pycparser in c:\users\socor\anaconda3\lib\site-pa
ckages (from cffi>=1.14; os_name == "nt" and implementation_name != "pypy"->trio~=0.17->selenium)
(2.20)
Installing collected packages: sparse, scikit-learn, slicer, packaging, shap, lightgbm, econml
  Attempting uninstall: scikit-learn
    Found existing installation: scikit-learn 0.23.1
    Uninstalling scikit-learn-0.23.1:
      Successfully uninstalled scikit-learn-0.23.1
  Attempting uninstall: packaging
    Found existing installation: packaging 20.4
    Uninstalling packaging-20.4:
```

```
      Successfully uninstalled packaging-20.4
Successfully installed econml-0.15.0 lightgbm-4.3.0 packaging-24.0 scikit-learn-1.3.2 shap-0.43.0
slicer-0.0.7 sparse-0.15.1
```

We need a list of all PCCs/force areas. let us scrape that list:

Having established that Selenium is capable of accessing the police.uk website, let's start building an ethical bot! Firstly, we accessed the https://police.uk/robots.txt (https://police.uk/robots.txt) page and found certain URLs needed to be disallowed. I decided to start by caching the robots.txt file so that my bot could refer to it without sending repeated requests to the site. My bot would then check URLs against those contained in the robot.txt file and would return a "robot.txt error" rather than crawl the forbidden URL:

It is customary to include a specific "user-agent" to identify your bot and make it possible for website administrators to contact you with concerns:

Data Collection

In [2]:
```python
from selenium.webdriver.chrome.options import Options
def establish_user_agent(user_agent, chromedriver_path):
    chrome_options = Options()
    chrome_options.add_argument(f"user-agent={user_agent}")
    return chrome_options
```

In [3]:
```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
def init_chrome_webdriver(chromedriver_path, chrome_options):
    chrome_options.add_argument("--no-sandbox")  # This parameter helps in avoiding unnecessary cras
    chrome_options.add_argument("--disable-gpu")  # Disables GPU hardware acceleration. If software
    chrome_options.add_argument("--log-level=3")  # This will only show fatal errors in the console
    service = Service(executable_path=chromedriver_path)
    driver = webdriver.Chrome(service=service, options=chrome_options)
    return driver
```

In [4]:
```python
import time
import json
from selenium.webdriver.common.by import By
def test_user_agent(driver, user_agent):
    driver.get("https://httpbin.org/user-agent")
    time.sleep(5)
    response_data = json.loads(driver.find_element(By.TAG_NAME, "body").text)
    echoed_user_agent = response_data["user-agent"]

    if echoed_user_agent != user_agent:
        print("User-Agent does not match the expected value. Quitting...")
        raise Exception("User-Agent does not match the expected value.")
```

In [5]:

```python
def is_target_disallowed(target, disallowed_dict):
    """
    Check if the target path matches any of the disallowed paths.

    :param target_path: The target path to check
    :param disallowed_paths: A dictionary of disallowed paths from robots.txtf files for each base_u
    :return: True if the target path is disallowed, False otherwise
    """
    # Extract base URL from the target
    parsed_url = urlparse(target)
    base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"

    # Retrieve the list of disallowed patterns for the base URL
    disallowed_patterns = disallowed_dict.get(base_url, [])

    # Normalize target path
    target_pattern = f'{parsed_url.path}?{parsed_url.query}'.rstrip("?")
    target_path = target_pattern.rstrip("/")

    for pattern in disallowed_patterns:
        # Normalize disallowed path
        pattern = pattern.rstrip("/")

        # Check if the target pattern starts with the disallowed pattern
        if target_path.startswith(pattern):
            return True
        # Checking for file extension disallowance, e.g., '*.aspx$'
        if pattern.endswith('$'):
            base_pattern = pattern[1:-1]
            if target_path.endswith(base_pattern):
                return True

    return False
```

In [6]:

```python
from urllib.parse import urlparse
import re
def establish_bot_permissions(driver, target, existing_disallowed=None):
    parsed_url = urlparse(target)
    base_url = f"{parsed_url.scheme}://{parsed_url.netloc}"

    # Initialize the dictionary if not provided
    if existing_disallowed is None:
        existing_disallowed = {}

    # If the base URL is already in the dictionary, return it
    if base_url in existing_disallowed:
        if is_target_disallowed(target, existing_disallowed):
            print('This URL is not allowed to be crawled in line with robots.txt')
            raise Exception(f"Target path {target} is disallowed.")
        else:
            print(f"{target} is not disallowed")
        return existing_disallowed

    # Navigate to relevant robots.txt file
    robots_url = f"{base_url}/robots.txt"
    driver.get(robots_url)
    time.sleep(1)

    # Scrape disallowed patterns
    robots_txt_content = driver.find_element(By.TAG_NAME, "body").text
    disallow_pattern = r"Disallow: ([^\n]+)"
    disallowed_paths = re.findall(disallow_pattern, robots_txt_content)
    existing_disallowed[base_url] = disallowed_paths

    if is_target_disallowed(target, existing_disallowed):
        print('This URL is not allowed to be crawled in line with robots.txt')
        raise Exception(f"Target path {target} is disallowed.")
    else:
            print(f"{target} is not disallowed")
    return existing_disallowed
```

In [7]:

```python
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

def get_force_areas(driver, target):
    try:
        driver.get(target)
        all_buttons = WebDriverWait(driver, 10).until(
            EC.presence_of_all_elements_located((By.CSS_SELECTOR, ".js-crime-stats-table-toggle"))
        )

        if len(all_buttons) > 1:
            toggle_button = all_buttons[1]  # Select the second button
            driver.execute_script("arguments[0].scrollIntoView(true);", toggle_button)
            toggle_button.click()
            time.sleep(2)
        else:
            print("Not enough buttons found.")

        tables = driver.find_elements(By.TAG_NAME, 'table')
        table = tables[-1]
        driver.execute_script("arguments[0].scrollIntoView(true);", table)
        rows = table.find_elements(By.TAG_NAME, 'tr')
        force_areas = []

        for row in rows:
            cells = row.find_elements(By.TAG_NAME, 'td')
            if cells:
                text = cells[0].text.strip()
                force_areas.append(text)
    except Exception as e:
        print(f"An error occurred while processing: {e}")
    return force_areas
```

In [8]:

```python
from selenium.webdriver.common.keys import Keys
def navigate_to_force_area_performance(driver, area, disallowed_patterns, force_area_urls={}):
    try:
        all_search_inputs = WebDriverWait(driver, 10).until(
            EC.visibility_of_all_elements_located((By.CSS_SELECTOR, "input[type='search'], input[nar
        )

        # Make sure there are at least two search bars
        if len(all_search_inputs) >= 2:
            search_input = all_search_inputs[1]  # Select the second search input
        else:
            raise Exception("Less than two search inputs found on the page.")

        search_input.click()
        # Clear the search field first in case there's any pre-filled text
        search_input.clear()
        # Enter the area name into the search field
        search_input.send_keys(area)
        # Search!
        search_input.send_keys(Keys.ENTER)  # Press Enter directly via Selenium

        time.sleep(1)
        #Check if this new page is disallowed
        target = driver.current_url
        disallowed_patterns = establish_bot_permissions(driver,target,disallowed_patterns)

        driver.get(target)
        time.sleep(1)
        print(f"Navigation to the {area} performance page is successful.")

    except Exception as e:
        print(f"An error occurred while navigating to the {area} performance page: {e}")
    return force_area_urls
```

In [9]:

```
def get_jurisdictions(driver, area, disallowed_patterns, force_area_jurisdictions={}):
    link = driver.find_elements(By.XPATH, "//a[.//h3[contains(@class, 'c-link-panel_title') and cont
    if len(link)<1:
        print("No data available")
        jurisdictions[area]={}
        return jurisdictions
    link = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.XPATH, "//a[.//h3[contains(@class, 'c-link-panel_ti
        )
    target = link.get_attribute('href')
    disallowed_patterns = establish_bot_permissions(driver,target,disallowed_patterns)
    driver.get(target)
    time.sleep(1)
    all_buttons = driver.find_elements(By.CSS_SELECTOR, ".js-crime-stats-table-toggle")
    if len(all_buttons) > 1:
        toggle_button = all_buttons[1]
        driver.execute_script("arguments[0].scrollIntoView(true);", toggle_button)
        toggle_button.click()
        time.sleep(1)
    else:
        print("Not enough buttons found.")
        jurisdictions[area]={}
        return jurisdictions

    tables = driver.find_elements(By.TAG_NAME, 'table')
    table = tables[2]
    driver.execute_script("arguments[0].scrollIntoView(true);", table)
    rows = table.find_elements(By.TAG_NAME, 'tr')
    force_area_jurisdictions = {}

    for row in rows:
        cells = row.find_elements(By.TAG_NAME, 'td')
        if cells:
            text = cells[0].text.strip()
            force_area_jurisdictions[text]=cells[1].text.strip()
    jurisdictions[area] = force_area_jurisdictions
    return jurisdictions
```

In [10]:

```
def get_force_area_finances(driver, area, disallowed_patterns,financial_reserves={}):
    navigate_to_force_area_performance(driver, area, disallowed_patterns)
    link = driver.find_elements(By.XPATH, "//a[.//h3[contains(@class, 'c-link-panel_title') and cont
    if len(link)<1:
        print("No data available")
        financial_reserves[area]={}
        return financial_reserves
    link = WebDriverWait(driver, 10).until(
            EC.visibility_of_element_located((By.XPATH, "//a[.//h3[contains(@class, 'c-link-panel_ti
        )
    target = link.get_attribute('href')
    disallowed_patterns = establish_bot_permissions(driver,target,disallowed_patterns)
    driver.get(target)
    time.sleep(1)
    all_buttons = driver.find_elements(By.CSS_SELECTOR, ".js-crime-stats-table-toggle")
    if len(all_buttons) > 1:
        toggle_button = all_buttons[0]
        driver.execute_script("arguments[0].scrollIntoView(true);", toggle_button)
        toggle_button.click()
        time.sleep(1)
    else:
        print("Not enough buttons found.")
        return financial_reserves

    tables = driver.find_elements(By.TAG_NAME, 'table')
    table = tables[-2]
    driver.execute_script("arguments[0].scrollIntoView(true);", table)
    rows = table.find_elements(By.TAG_NAME, 'tr')

    for row in rows:
        cells = row.find_elements(By.TAG_NAME, 'td')
        if cells:
            year = cells[0].text.strip()
            financial_reserves[area][year]['General fund']=cells[1].text.strip()
            financial_reserves[area][year]['Earmarked reserves']=cells[2].text.strip()
            financial_reserves[area][year]['Total resource reserves']=cells[3].text.strip()
            financial_reserves[area][year]['Capital reserves']=cells[4].text.strip()
    return financial_reserves
# {force_area:{Mar 2018: {General Fund: 10000, Earmarked Reserves: 10000, Total Resource Reserves: 1
# {force_area_keys:{Year_keys:{Fund_type_keys:Values}}}
```

What follows is the webscraping script- remember to recreate this script's output, you must have first downloaded the relevant chromedriver for your machine from https://googlechromelabs.github.io/chrome-for-testing/#stable (https://googlechromelabs.github.io/chrome-for-testing/#stable), and provide the path to your own version of the chromedriver where prompted in the script. You may also wish to use your own user-agent. It is recommended that your user-agent contains a (+mailto:emailaddress) string so that any crawling of the bot that raises concerns with the service provider can be mediated by them reaching out to you.

In [23]:

```
# Setup User-Agent
user_agent = "FriendlyUniStudentResearcher/1.0 (+mailto:soc204@exeter.ac.uk)"

#Provide the path to your own version of the chromedriver
chromedriver_path = r"C:\Users\socor\Downloads\chromedriver-win64\chromedriver-win64\chromedriver.ex

chrome_options = establish_user_agent(user_agent, chromedriver_path)

# Initialize the WebDriver (assuming Chrome)
driver = init_chrome_webdriver(chromedriver_path,chrome_options)

# Set target URL
target = 'https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/financial-reserv

try:
    # Navigate to a website that echoes back the user-agent
    test_user_agent(driver, user_agent)

    # Navigate to target website robots.txt and save the disallowed patterns
    disallowed_patterns = establish_bot_permissions(driver, target)

    # Collect the names of Force areas for which data is available
    Force_Areas = get_force_areas(driver, target)

    target = 'https://www.police.uk/pu/performance/'
    disallowed_patterns = establish_bot_permissions(driver, target, disallowed_patterns)
    driver.get(target)
    force_area_urls = {}
    jurisdictions = {}
    financial_reserves = {}
    Periods = ('Mar 2011', 'Mar 2012', 'Mar 2013', 'Mar 2014', 'Mar 2015', 'Mar 2016', 'Mar 2017',
    Reserves = ('General fund','Earmarked reserves', 'Total resource reserves', 'Capital reserves')
    for area in Force_Areas:
        period_dict={}
        for period in Periods:
            reserves_dict={}
            for reserve_type in Reserves:
                reserves_dict[reserve_type] = None
                period_dict[period] = reserves_dict
                financial_reserves[area] = period_dict
    # Target each force area's performance data
    for area in Force_Areas[:-1]:
        force_area_urls = navigate_to_force_area_performance(driver, area, disallowed_patterns, fore
        jurisdictions = get_jurisdictions(driver, area, disallowed_patterns)
        #get force area's historical financial reserves
        financial_reserves = get_force_area_finances(driver, area, disallowed_patterns,financial_res

        driver.get('https://www.police.uk/pu/performance/')
        time.sleep(2)

    time.sleep(10)

except Exception as e:
    print(f"An error occurred: {e}")
finally:
    # Close the browser
    driver.quit()
```

```
https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/financial-reserves/
(https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/financial-reserves/)
is not disallowed
https://www.police.uk/pu/performance/ (https://www.police.uk/pu/performance/) is not disallowed
https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/performance-avon-some
rset/?tc=AN004 (https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/perfo
rmance-avon-somerset/?tc=AN004) is not disallowed
Navigation to the Avon and Somerset Constabulary performance page is successful.
https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/compare-your-area/?tc
=AN004 (https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/compare-your-
area/?tc=AN004) is not disallowed
https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/performance-avon-some
rset/?tc=AN004 (https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/perfo
rmance-avon-somerset/?tc=AN004) is not disallowed
Navigation to the Avon and Somerset Constabulary performance page is successful.
https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/financial-reserves/?t
c=AN004 (https://www.police.uk/pu/your-area/avon-somerset-constabulary/performance/financial-re
serves/?tc=AN004) is not disallowed
https://www.police.uk/pu/your-area/bedfordshire-police/performance/performance-bedfordshire-pol
```

Data Cleaning

The webpage states that data is not available for "City of London Police" force area, so we'll add that force area manually. Similarly, most of the desired data was unavailable for the aggregate "Total England & Wales", so we need to add empty dictionaries to represent missing values in our data for those two "force areas".

In [12]:
```
Force_Areas.append("City of London Police")
```

We need to store that data in a pandas series to unlock better functionality. The idea is to get force area level data on financial reserves over the period since records begin and average crime rate for each force area last year.

In [167]:

```
import pandas as pd
import numpy as np

force_avg_crime_rate = {}
force_jurisdictions = {}
financial_data = {}

# Process average crime rates and jurisdictions
for area in Force_Areas:
    force_avg_crime_rate[area] = jurisdictions.get(area, {}).get('Force average', np.nan)
    force_jurisdictions[area] = [j for j in jurisdictions.get(area, {}) if j != 'Force average'] if

# Create Pandas Series
Force_Crime_Rates = pd.Series(force_avg_crime_rate, name='Average Crime Rate')
Force_Jurisdictions = pd.Series(force_jurisdictions, name='Jurisdictions')

# Create DataFrame from Series
ForceAreas = pd.DataFrame({'Average Crime Rate': Force_Crime_Rates, 'Jurisdictions': Force_Jurisdic

# Process financial reserves
for area in Force_Areas:
    area_data = financial_reserves.get(area, {})
    for period in Periods:
        for fund in Reserves:
            key = f"{period} {fund}"
            value = area_data.get(period, {}).get(fund, np.nan)
            financial_data.setdefault(key, {})[area] = value

# Add financial data to DataFrame
for key, values in financial_data.items():
    ForceAreas[key] = pd.Series(values)

ForceAreas.index.name = 'Force Area'
ForceAreas
```

Force Area									
Warwickshire Police	71.41	[Rugby, South Warwickshire, North Warwickshire...	£2.0m	£18.4m	£20.4m	£0.0m	£2.0m	£22.0m	£24.0m
West Mercia Police	71.74	[Shropshire, Herefordshire, South Worcestershi...	£6.2m	£25.9m	£32.1m	£2.7m	£6.2m	£31.6m	£37.8m
West Midlands Police	119.7	[Dudley, Solihull, Walsall, Coventry, Sandwell...	£8.0m	£71.1m	£79.1m	£39.9m	£12.0m	£90.5m	£102.5m
West Yorkshire Police	132.34	[Kirklees, Calderdale, Wakefield, Bradford, Le...	£23.9m	£6.5m	£30.4m	£1.3m	£30.7m	£11.0m	£41.6m
Wiltshire Police	59.43	[Wiltshire County, Swindon]	£7.7m	£9.0m	£16.7m	£0.0m	£8.9m	£11.7m	£20.7m
Total England &									

In [169]:

```
def convert_currency(value):
    # Check if the value is a string
    if isinstance(value, str):
        # Remove the pound sign and any other non-numeric characters except for '.'
        value = value.replace('£', '').replace('m', '').strip()
        # Convert to float and then scale if 'm' was in the original string
        try:
            value = float(value)
            value *= 1000000  # Convert millions to a plain numeric value
        except ValueError:
            return None
    return value
```

In [170]:
```
def string_to_float(string):
    value = 1
    # Check if the value is a string
    if isinstance(string, str):
        try:
            value = float(string)
        except ValueError:
            return np.nan
        return value
```

In [171]:
```
for finances in ForceAreas.iloc[:,2:].columns:
    ForceAreas[finances]=ForceAreas[finances].apply(convert_currency)
ForceAreas
```

Out[171]:

Force Area	Average Crime Rate	Jurisdictions	Mar 2011 General fund	Mar 2011 Earmarked reserves	Mar 2011 Total resource reserves	Mar 2011 Capital reserves	Mar 2012 General fund	Mar 2012 Earmarked reserves
Avon and Somerset Constabulary	83.24	[Bath & North East Somerset, South Gloucesters...	6700000.0	25600000.0	32200000.0	2100000.0	7500000.0	29600000.0
Bedfordshire Police	73.8	[Central Bedfordshire, Bedford, Luton]	2900000.0	6500000.0	9400000.0	400000.0	2900000.0	6600000.0
Cambridgeshire Constabulary	84.51	[East Cambridgeshire, South Cambridgeshire, Hu...	4800000.0	13300000.0	18100000.0	18200000.0	7000000.0	19100000.0

In [172]:
```
ForceAreas['Average Crime Rate']=ForceAreas['Average Crime Rate'].str.strip().apply(string_to_float
ForceAreas
```

Warwickshire Police	71.41	[Rugby, South Warwickshire, North Warwickshire...	2000000.0	18400000.0	20400000.0	0.0	2000000.0	22000000.0
West Mercia Police	71.74	[Shropshire, Herefordshire, South Worcestershi...	6200000.0	25900000.0	32100000.0	2700000.0	6200000.0	31600000.0
West Midlands Police	119.70	[Dudley, Solihull, Walsall, Coventry, Sandwell...	8000000.0	71100000.0	79100000.0	39900000.0	12000000.0	90500000.0
West Yorkshire Police	132.34	[Kirklees, Calderdale, Wakefield, Bradford, Le...	23900000.0	6500000.0	30400000.0	1300000.0	30700000.0	11000000.0
Wiltshire Police	59.43	[Wiltshire County, Swindon]	7700000.0	9000000.0	16700000.0	0.0	8900000.0	11700000.0
Total England &								

In [173]:
```
for period in Periods:
    newColumn = str(period+' Wealth')
    ForceAreas[newColumn] = ForceAreas[str(period+ ' Total resource reserves')] + ForceAreas[str(pe
ForceAreas['Average Wealth']=ForceAreas.iloc[:,-8:].mean(axis=1)
ForceAreas['Max Wealth']=ForceAreas.iloc[:,-9:-1].max(axis=1)
mean_row=ForceAreas.mean(skipna=True)
ForceAreas.loc['Mean Force Area'] = mean_row
ForceAreas
```

Mar 2012 Total resource reserves	Mar 2012 Capital reserves	...	Mar 2011 Wealth	Mar 2012 Wealth	Mar 2013 Wealth	Mar 2014 Wealth	Mar 2015 Wealth	Mar 2016 Wealth	Mar 2017 Wealth	Mar 20 Wea
3.710000e+07	2.400000e+06	...	3.430000e+07	3.950000e+07	4.410000e+07	4.980000e+07	6.020000e+07	5.130000e+07	44200000.0	3.600000e-
9.500000e+06	4.000000e+05	...	9.800000e+06	9.900000e+06	1.320000e+07	1.800000e+07	1.780000e+07	1.370000e+07	13800000.0	9.500000e-
2.610000e+07	2.630000e+07	...	3.630000e+07	5.240000e+07	4.410000e+07	5.250000e+07	2.770000e+07	2.830000e+07	29700000.0	2.440000e-

We also want a dataframe that combines jurisdictions with their crime rates (excluding force area averages) so that we can see the worst 5 jurisdictions for crime rate and top 5 jurisdictions for crime rate. From that we need a dataframe linking each jurisdiction a force area so that we can identify which force areas have the most success and whether reserve resources are an effective predictor of reduced crime rate.

In [193]:
```
data = []
for area, jurisdiction1 in jurisdictions.items():
    for jurisdiction2, crime_rate in jurisdiction1.items():
        data.append({'Force Area': area, 'Jurisdiction': jurisdiction2, 'Crime Rate': crime_rate})
jurisdiction_df = pd.DataFrame(data)
JurisdictionCrimes = jurisdiction_df[jurisdiction_df['Jurisdiction'] != 'Force average'].copy()
JurisdictionCrimes.set_index('Jurisdiction',inplace=True)
JurisdictionCrimes
```

Out[193]:

Jurisdiction	Force Area	Crime Rate
Bath & North East Somerset	Avon and Somerset Constabulary	64.78
South Gloucestershire	Avon and Somerset Constabulary	65.18
Somerset	Avon and Somerset Constabulary	70.55
North Somerset	Avon and Somerset Constabulary	71.27
Bristol	Avon and Somerset Constabulary	118.47
...	...	...
Wakefield	West Yorkshire Police	138.20
Bradford	West Yorkshire Police	139.58
Leeds	West Yorkshire Police	140.83
Wiltshire County	Wiltshire Police	51.05
Swindon	Wiltshire Police	77.57

268 rows × 2 columns

In [194]:
```
JurisdictionCrimes['Crime Rate']=JurisdictionCrimes['Crime Rate'].apply(string_to_float)
JurisdictionCrimes
```

Out[194]:

Jurisdiction	Force Area	Crime Rate
Bath & North East Somerset	Avon and Somerset Constabulary	64.78
South Gloucestershire	Avon and Somerset Constabulary	65.18
Somerset	Avon and Somerset Constabulary	70.55
North Somerset	Avon and Somerset Constabulary	71.27
Bristol	Avon and Somerset Constabulary	118.47
...	...	...
Wakefield	West Yorkshire Police	138.20
Bradford	West Yorkshire Police	139.58
Leeds	West Yorkshire Police	140.83
Wiltshire County	Wiltshire Police	51.05
Swindon	Wiltshire Police	77.57

268 rows × 2 columns

In [209]:
```
RankedJurisdictions=JurisdictionCrimes.sort_values('Crime Rate', ascending=True)
RankedJurisdictions.loc['Mean Jurisdiction','Crime Rate']=RankedJurisdictions['Crime Rate'].mean()
```

Out[209]:
```
Force Area        NaN
Crime Rate    82.6449
Name: Mean Jurisdiction, dtype: object
```

In [218]:
```
Top5Jurisdictions = RankedJurisdictions.iloc[:5,:]
RelativeComparison_Top = Top5Jurisdictions.copy()
RelativeComparison_Top.loc['Mean Jurisdiction'] = RankedJurisdictions.loc['Mean Jurisdiction','Crime
RelativeComparison_Top.loc['Mean Jurisdiction','Force Area'] = np.nan
RelativeComparison_Top
```

Out[218]:

Jurisdiction	Force Area	Crime Rate
Isles of Scilly	Devon & Cornwall Police	24.110000
South Devon & Dartmoor	Devon & Cornwall Police	41.470000
Broadland	Norfolk Constabulary	41.660000
Suffolk Coastal	Suffolk Constabulary	41.900000
Wealden	Sussex Police	43.220000
Mean Jurisdiction	NaN	82.644888

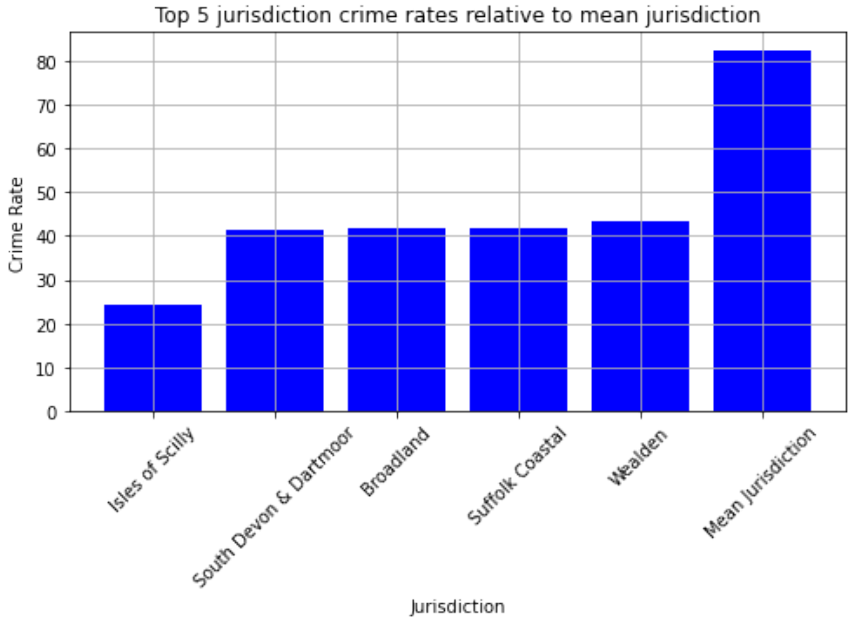
In [222]:
```
Bottom5Jurisdictions = RankedJurisdictions.iloc[-6:,:]
RelativeComparison_Bottom = Bottom5Jurisdictions.copy()
RelativeComparison_Bottom.loc['Mean Jurisdiction'] = RankedJurisdictions.loc['Mean Jurisdiction','Cr
RelativeComparison_Bottom.loc['Mean Jurisdiction','Force Area'] = np.nan
RelativeComparison_Bottom
```

Out[222]:

Jurisdiction	Force Area	Crime Rate
Leeds	West Yorkshire Police	140.830000
Liverpool	Merseyside Police	141.620000
Hartlepool	Cleveland Police	153.470000
Manchester	Greater Manchester Police	181.500000
Middlesbrough	Cleveland Police	183.780000
Mean Jurisdiction	NaN	82.644888

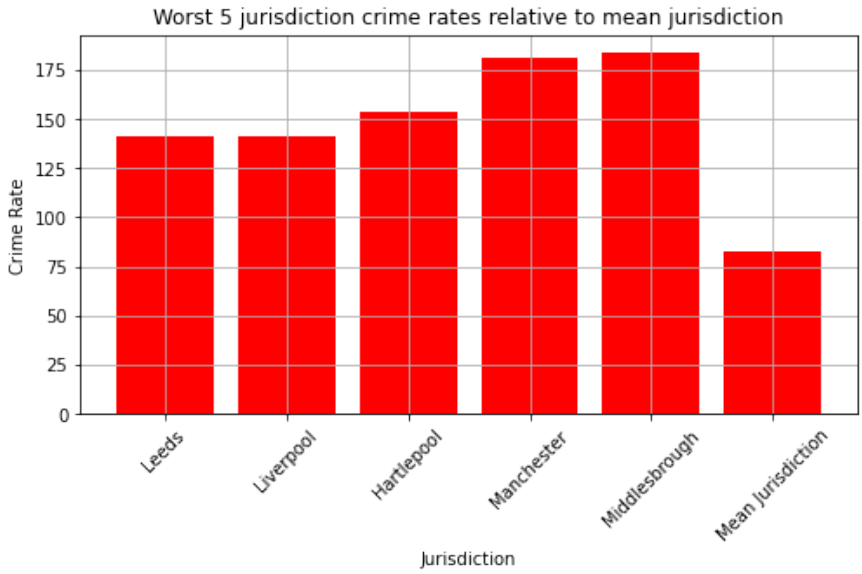
Data Analysis

In [252]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(8,4))
plt.bar(RelativeComparison_Top.index, RelativeComparison_Top['Crime Rate'], color = 'blue')
plt.title('Top 5 jurisdiction crime rates relative to mean jurisdiction')
plt.xlabel('Jurisdiction')
plt.ylabel('Crime Rate')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



In [253]:

```
plt.figure(figsize=(8,4))
plt.bar(RelativeComparison_Bottom.index, RelativeComparison_Bottom['Crime Rate'], color = 'red')
plt.title('Worst 5 jurisdiction crime rates relative to mean jurisdiction')
plt.xlabel('Jurisdiction')
plt.ylabel('Crime Rate')
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```

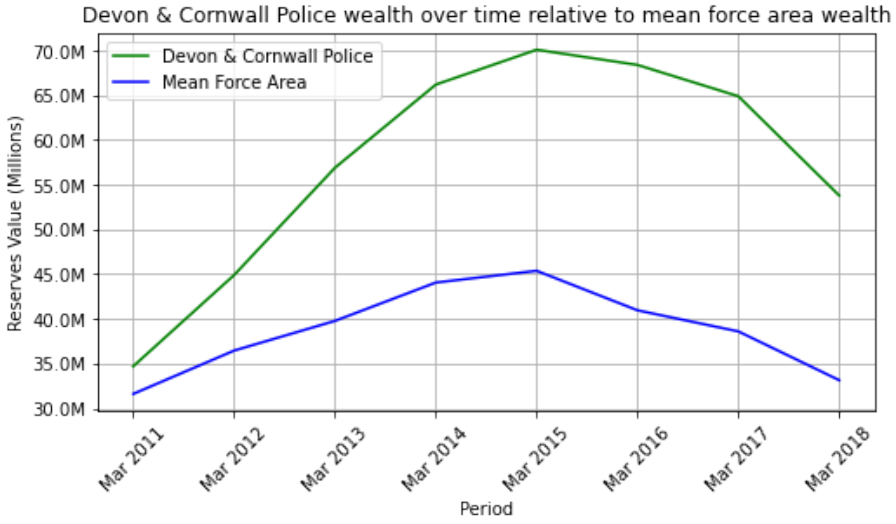


Case Study: Best Force Area

In [249]:

```
import matplotlib.ticker as ticker
TopForceArea = Top5Jurisdictions['Force Area'].iloc[1]
print('Best Force Area: ' + TopForceArea)
WealthTime = ForceAreas.iloc[:,-10:-2]
TopForceWealthTime=WealthTime.loc[TopForceArea]
x_axis = [i[:8] for i in TopForceWealthTime.index[:]]
plt.figure(figsize=(8,4))
plt.plot(x_axis,TopForceWealthTime, color='green',label = TopForceArea)
plt.plot(x_axis,ForceAreas.loc['Mean Force Area'].iloc[-10:-2],color = 'blue', label = 'Mean Force A
plt.xticks(rotation=45)
scale_factor = 1e6
ticks_loc = plt.gca().get_yticks().tolist()
plt.gca().yaxis.set_major_locator(ticker.FixedLocator(ticks_loc))
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: '{:0.1f}M'.format(x/scale_fac
plt.title(str(TopForceArea + ' wealth over time relative to mean force area wealth'))
plt.xlabel('Period')
plt.ylabel('Reserves Value (Millions)')
plt.legend()
plt.grid(True)
plt.show()
```

```
Best Force Area: Devon & Cornwall Police
```

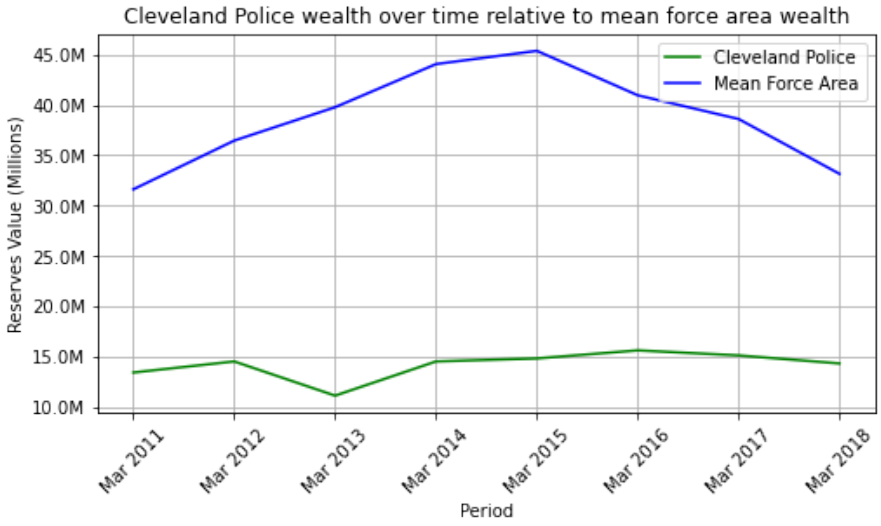


Case Study: Worst Force Area

In [247]:

```
WorstForceArea = Bottom5Jurisdictions['Force Area'].iloc[-2]
print('Worst Force Area: ' + WorstForceArea)
WealthTime = ForceAreas.iloc[:,-10:-2]
WorstForceWealthTime=WealthTime.loc[WorstForceArea]
x_axis = [i[:8] for i in WorstForceWealthTime.index[:]]
plt.figure(figsize=(8,4))
plt.plot(x_axis,WorstForceWealthTime, color='green', label = WorstForceArea)
plt.plot(x_axis,ForceAreas.loc['Mean Force Area'].iloc[-10:-2], color= 'blue', label = 'Mean Force A
plt.xticks(rotation=45)
scale_factor = 1e6
ticks_loc = plt.gca().get_yticks().tolist()
plt.gca().yaxis.set_major_locator(ticker.FixedLocator(ticks_loc))
plt.gca().yaxis.set_major_formatter(ticker.FuncFormatter(lambda x, _: '{:0.1f}M'.format(x/scale_fac
plt.title(str(WorstForceArea + ' wealth over time relative to mean force area wealth'))
plt.xlabel('Period')
plt.ylabel('Reserves Value (Millions)')
plt.legend()
plt.grid(True)
plt.show()
```

```
Worst Force Area: Cleveland Police
```



How well does wealth held in reserves predict a force area's crime rate?

In [257]:

```
from econml.dml import CausalForestDML
from sklearn.ensemble import GradientBoostingRegressor

# Define the causal forest model
causal_forest = CausalForestDML(
    model_y=GradientBoostingRegressor(),
    model_t=GradientBoostingRegressor(),
    discrete_treatment=False,
    random_state=123
)

# Fit the model
X = pd.DataFrame()  # No covariates
T = ForceAreas['Average Wealth']  # Treatment
Y = ForceAreas['Average Crime Rate']  # Outcome
causal_forest.fit(Y, T, X=X)

# Estimate the causal effect
effects = causal_forest.effect(X)
print("Estimated Causal Effects:\n", effects)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-257-5a8a4cc961e3> in <module>
     14 T = ForceAreas['Average Wealth']  # Treatment
     15 Y = ForceAreas['Average Crime Rate']  # Outcome
---> 16 causal_forest.fit(Y, T, X=X)
     17
     18 # Estimate the causal effect

~\anaconda3\lib\site-packages\econml\dml\causal_forest.py in fit(self, Y, T, X, W, sample_weight,
groups, cache_values, inference)
    852         if X is None:
    853             raise ValueError("This estimator does not support X=None!")
--> 854         return super().fit(Y, T, X=X, W=W,
    855                            sample_weight=sample_weight, groups=groups,
    856                            cache_values=cache_values,

~\anaconda3\lib\site-packages\econml\dml\_rlearner.py in fit(self, Y, T, X, W, sample_weight, freq
_weight, sample_var, groups, cache_values, inference)
    420         """
    421         # Replacing fit from _OrthoLearner, to enforce Z=None and improve the docstring
--> 422         return super().fit(Y, T, X=X, W=W,
    423                            sample_weight=sample_weight, freq_weight=freq_weight, sample_va
r=sample_var, groups=groups,
    424                            cache_values=cache_values,

~\anaconda3\lib\site-packages\econml\_cate_estimator.py in call(self, Y, T, inference, *args, **kw
args)
    129                 inference.prefit(self, Y, T, *args, **kwargs)
    130             # call the wrapped fit method
--> 131             m(self, Y, T, *args, **kwargs)
    132             self._postfit(Y, T, *args, **kwargs)
    133             if inference is not None:

~\anaconda3\lib\site-packages\econml\_ortho_learner.py in fit(self, Y, T, X, W, Z, sample_weight,
freq_weight, sample_var, groups, cache_values, inference, only_final, check_input)
    753             "is not supported when treatment is discrete"
    754         if check_input:
--> 755             Y, T, Z, sample_weight, freq_weight, sample_var, groups = check_input_arrays(
    756                 Y, T, Z, sample_weight, freq_weight, sample_var, groups)
    757             X, = check_input_arrays(

~\anaconda3\lib\site-packages\econml\utilities.py in check_input_arrays(validate_len, force_all_fi
nite, dtype, *args)
    576     for i, arg in enumerate(args):
    577         if np.ndim(arg) > 0:
--> 578             new_arg = check_array(arg, dtype=dtype, ensure_2d=False, accept_sparse=True,
    579                                   force_all_finite=force_all_finite)
    580             if not force_all_finite:

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in check_array(array, accept_sparse, acc
ept_large_sparse, dtype, order, copy, force_all_finite, ensure_2d, allow_nd, ensure_min_samples, e
nsure_min_features, estimator, input_name)
    955
    956         if force_all_finite:
--> 957             _assert_all_finite(
    958                 array,
    959                 input_name=input_name,

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in _assert_all_finite(X, allow_nan, msg_
dtype, estimator_name, input_name)
    120         return
    121
--> 122     _assert_all_finite_element_wise(
    123         X,
    124         xp=xp,

~\anaconda3\lib\site-packages\sklearn\utils\validation.py in _assert_all_finite_element_wise(X, x
p, allow_nan, msg_dtype, estimator_name, input_name)
    169                 "#estimators-that-handle-nan-values"
    170             )
--> 171         raise ValueError(msg_err)
    172
```

```
    173
```

```
ValueError: Input contains NaN.
```

https://github.com/SOCStudentUoE/BEE2041-Empirical-Assignment (https://github.com/SOCStudentUoE/BEE2041-Empirical-Assignment)