FDS-Project-Code

February 13, 2026

Please note:

1. This document is a template for your project code notebook, and gives some hints, but is not an exemplar
2. Retain the 4 top level headings ("Imports", "Data", "Exploratory Data Analysis", "Data Communication and Analysis")
3. Edit, remove or insert other cells as you wish
4. Before submission, **restart and run the whole Notebook from start to finish**, so that the numbers by the code cells start at `[1]` and go up `[2]`, `[3]`, …
5. Then save as PDF using **File→Save and Export Notebook as→PDF**

1 Imports

You can import any packages you would like, but make sure you know what they are doing.

[1]:
```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
mpl.rcParams['font.size'] = 8    # Set the font size to 8pt
mpl.rcParams['figure.dpi'] = 200 # Set the dots per inch to a reasonable␣
 ↪compromise between resolution and size
```

2 Data

Code to load and process the data.

[2]:
```
# Load data
simd = pd.read_csv('datasets/simd.csv')
```

3 Exploratory Data Analysis

Exploratory Data Analysis of your chosen dataset, which should demonstrate that you have understood the structure of the data, identified anomalous or missing data, and used visualisation and descriptive statistics to explore possible relationships in the data.

[3]:
```
simd.head()
```

[3]:

	Data_Zone	Intermediate_Zone	Council_area	Total_population	\
0	S01006506	Culter	Aberdeen City	904	
1	S01006507	Culter	Aberdeen City	830	
2	S01006508	Culter	Aberdeen City	694	
3	S01006509	Culter	Aberdeen City	573	
4	S01006510	Culter	Aberdeen City	676	

	Working_age_population_revised	Income_rate	Income_count	Employment_rate	\
0	605	7%	60	7%	
1	491	7%	60	5%	
2	519	5%	30	3%	
3	354	5%	30	6%	
4	414	10%	70	7%	

	Employment_count	CIF	…	drive_secondary	PT_GP	PT_Post	PT_retail	\
0	40	60	…	10.8	8.4	6.0	5.7	
1	25	40	…	11.5	8.3	7.3	6.8	
2	15	45	…	11.5	7.9	5.8	5.3	
3	20	65	…	10.8	7.4	8.3	8.4	
4	30	75	…	10.6	5.1	6.6	6.6	

	crime_count	crime_rate	overcrowded_count	nocentralheat_count	\
0	8	89	87	10	
1	4	48	85	4	
2	4	58	31	8	
3	*	*	42	6	
4	12	178	50	7	

	overcrowded_rate	nocentralheat_rate
0	10%	1%
1	10%	0%
2	5%	1%
3	7%	1%
4	9%	1%

[5 rows x 36 columns]

We can see that each row corresponds to a Data_Zone, and the columns contain various statistics for each Zone.

[4]:
```
simd.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6976 entries, 0 to 6975
Data columns (total 36 columns):
 #   Column                           Non-Null Count  Dtype
---  ------                           --------------  -----
 0   Data_Zone                        6976 non-null   object
 1   Intermediate_Zone                6976 non-null   object
 2   Council_area                     6976 non-null   object
 3   Total_population                 6976 non-null   int64
 4   Working_age_population_revised   6976 non-null   int64
 5   Income_rate                      6976 non-null   object
 6   Income_count                     6976 non-null   int64
 7   Employment_rate                  6976 non-null   object
 8   Employment_count                 6976 non-null   int64
 9   CIF                              6976 non-null   object
 10  ALCOHOL                          6976 non-null   object
 11  DRUG                             6976 non-null   int64
 12  SMR                              6976 non-null   int64
 13  DEPRESS                          6976 non-null   object
 14  LBWT                             6976 non-null   object
 15  EMERG                            6976 non-null   int64
 16  Attendance                       6976 non-null   object
 17  Attainment                       6976 non-null   object
 18  Noquals                          6976 non-null   int64
 19  NEET                             6976 non-null   object
 20  HESA                             6976 non-null   object
 21  drive_petrol                     6976 non-null   float64
 22  drive_GP                         6976 non-null   float64
 23  drive_PO                         6976 non-null   float64
 24  drive_primary                    6976 non-null   float64
 25  drive_retail                     6976 non-null   float64
 26  drive_secondary                  6976 non-null   float64
 27  PT_GP                            6976 non-null   float64
 28  PT_Post                          6976 non-null   float64
 29  PT_retail                        6976 non-null   float64
 30  crime_count                      6976 non-null   object
 31  crime_rate                       6976 non-null   object
 32  overcrowded_count                6976 non-null   int64
 33  nocentralheat_count              6976 non-null   int64
 34  overcrowded_rate                 6976 non-null   object
 35  nocentralheat_rate               6976 non-null   object
dtypes: float64(9), int64(10), object(17)
memory usage: 1.9+ MB
```

It looks like the data is in a wide format. There are some columns (e.g. `Income_rate`), which are objects, even though we might think they should be numbers. Looking at the output above, it

will be because they are expressed as percentage strings. Depending on what analysis we do, we might want to convert these columns to numeric ones, in the Data section, in the iteration of this notebook.

[5]:
```
simd.describe()
```

[5]:

	Total_population	Working_age_population_revised	Income_count
count	6976.000000	6976.000000	6976.000000
mean	766.571101	493.098337	94.598624
std	188.267392	151.225436	76.077188
min	0.000000	0.000000	0.000000
25%	635.000000	397.000000	35.000000
50%	753.000000	474.000000	75.000000
75%	877.000000	563.000000	135.000000
max	3302.000000	2917.000000	555.000000

	Employment_count	DRUG	SMR	EMERG	Noquals
count	6976.000000	6976.000000	6976.000000	6976.000000	6976.000000
mean	52.990252	96.548739	99.778670	100.752150	102.006164
std	42.075838	151.610455	44.981516	34.361399	55.676766
min	0.000000	0.000000	0.000000	20.000000	3.000000
25%	20.000000	0.000000	71.000000	74.000000	58.000000
50%	40.000000	42.000000	93.000000	95.000000	93.000000
75%	75.000000	129.000000	121.000000	122.000000	140.000000
max	325.000000	1864.000000	950.000000	324.000000	353.000000

	drive_petrol	drive_GP	drive_PO	drive_primary	drive_retail
count	6976.000000	6976.000000	6976.000000	6976.000000	6976.000000
mean	3.681035	3.402838	2.737686	2.516170	5.238733
std	2.926468	2.786016	1.616610	3.427299	6.131718
min	0.600000	0.600000	0.600000	0.700000	0.700000
25%	2.100000	1.900000	1.700000	1.600000	2.800000
50%	2.950000	2.700000	2.400000	2.100000	4.000000
75%	4.100000	3.900000	3.200000	2.800000	5.900000
max	63.500000	87.800000	17.300000	186.100000	190.000000

	drive_secondary	PT_GP	PT_Post	PT_retail
count	6976.000000	6976.000000	6976.000000	6976.000000
mean	6.110751	10.266972	8.569696	13.532010
std	5.179672	6.182297	4.477807	10.660695
min	1.200000	1.600000	1.900000	1.900000
25%	3.700000	6.300000	5.600000	8.000000
50%	4.800000	8.800000	7.500000	11.200000
75%	6.700000	12.300000	10.200000	16.100000
max	116.100000	108.800000	40.300000	190.000000

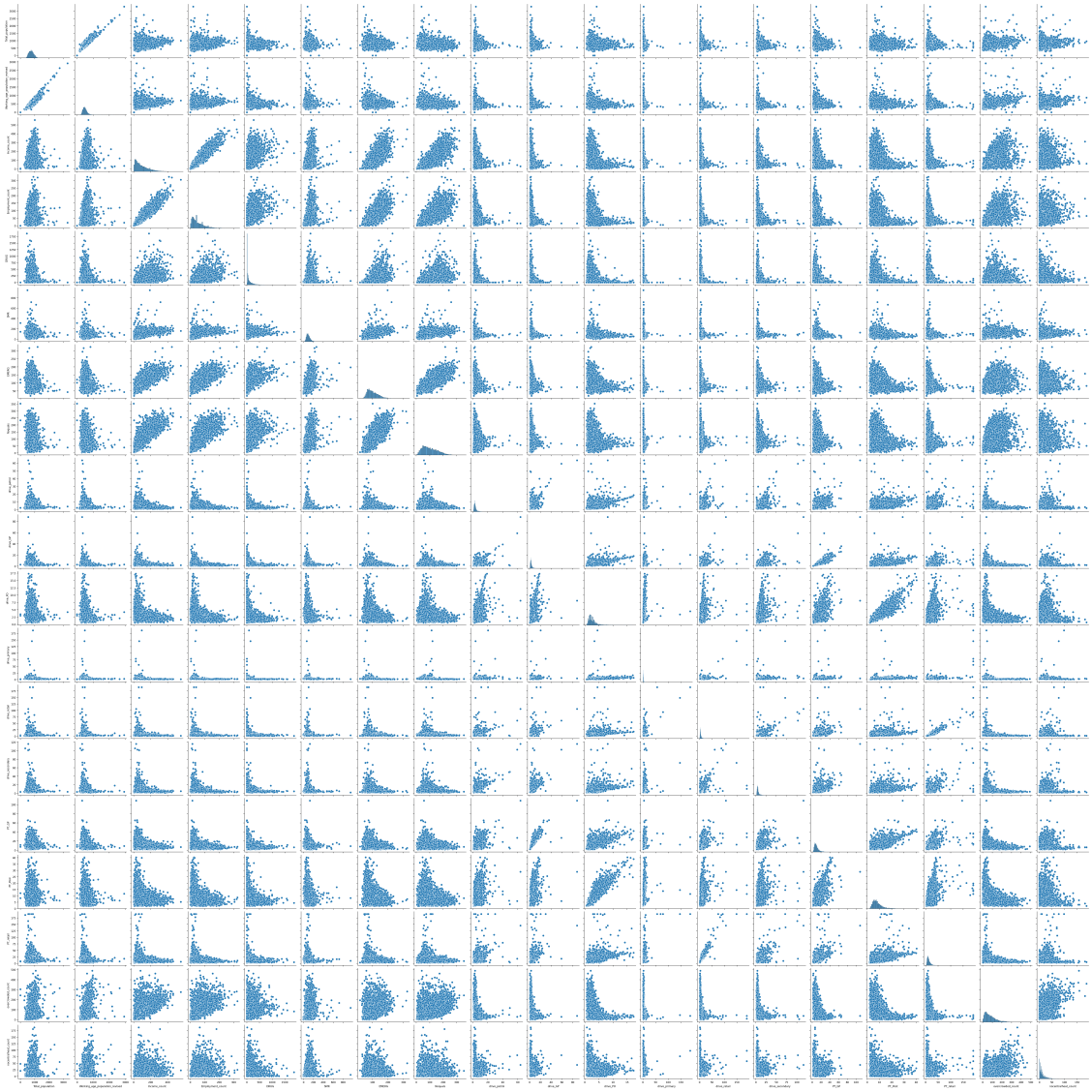
	overcrowded_count	nocentralheat_count

count	6976.000000	6976.000000
mean	82.332569	13.430619
std	63.921332	16.800241
min	0.000000	0.000000
25%	35.000000	3.000000
50%	66.500000	8.000000
75%	112.000000	17.000000
max	490.000000	187.000000

We can see some summary stats for the numeric items above. We should check the max and min values for the columns we're using make sense.

[6]:
```
sns.pairplot(data=simd)
```

[6]:
```
<seaborn.axisgrid.PairGrid at 0x7f2f03481490>
```

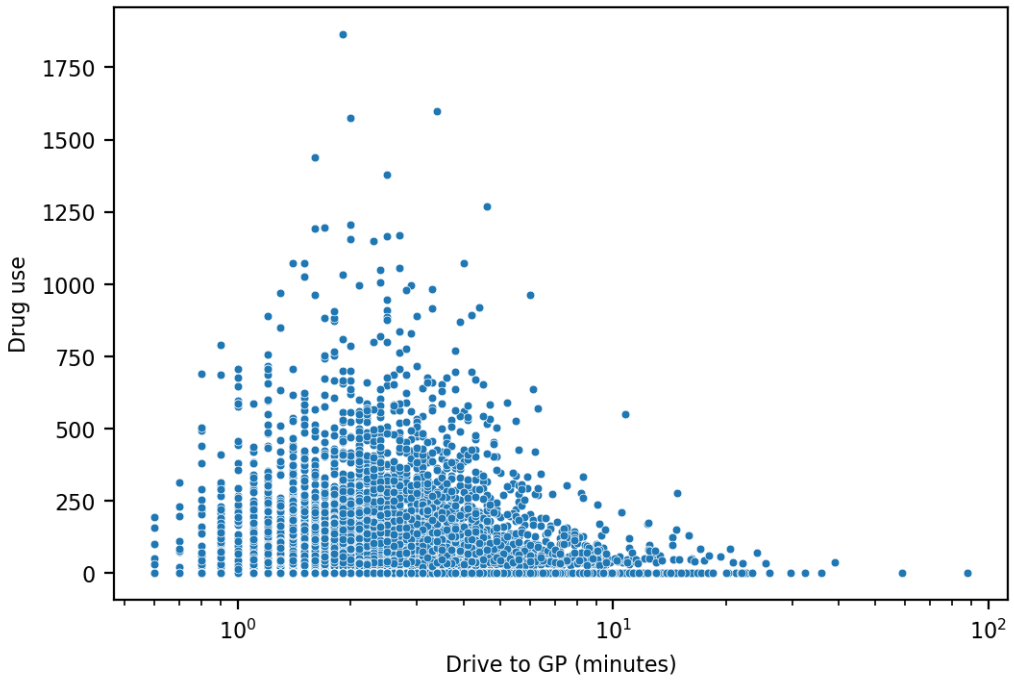


A pairplot is often helpful in understanding numeric data - as this is in the EDA section, we don't mind if it's not easily legible - but you make want to reduced the number of columns shown so you can see what the labels are.

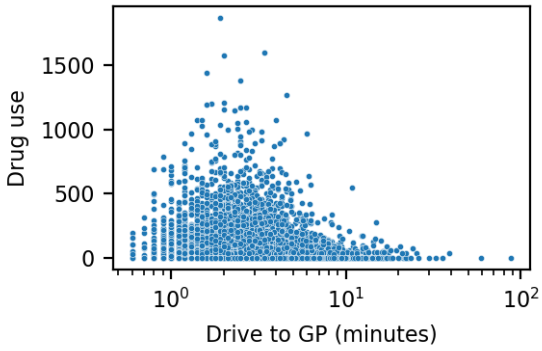
4 Data communication and analysis

The visualisation(s), tables and any statistical analyses or application of ML to the data, as presented in your report.

[7]:
```
# If you make the plot wider than 6in, it will have to be shrunk to fit in the␣
↪document, and the font size will become too small
plt.figure(figsize=(6,4))
sns.scatterplot(data=simd, x='drive_GP', y='DRUG', s=10)
plt.xscale('log')
plt.xlabel('Drive to GP (minutes)')
plt.ylabel('Drug use')
plt.savefig('example1-large.png') # Save the figure as a PNG
```



[8]:
```
plt.figure(figsize=(3,2))
sns.scatterplot(data=simd, x='drive_GP', y='DRUG', s=5)
plt.xscale('log')
plt.xlabel('Drive to GP (minutes)')
plt.ylabel('Drug use')
plt.tight_layout()
plt.savefig('example1-small.png') # Save the figure as a PNG
```



You can export tables to LaTeX, with the Pandas styler object.

[9]:
```
# This is just to make a smaller table to output
simd_output = simd.copy().head()
# Table headings are better without underscores - and LaTeX doesn't like␣
 ↪underscores
simd_output.columns =  simd_output.columns.str.replace('_', ' ')
# Save to a LaTeX file
(
    simd_output[['Data Zone', 'Council area', 'Total population', 'Income␣
 ↪rate']]
        .style # Styler object
        .format(escape='latex') # Escape LaTeX characters
        .hide(axis=0) # Hide the index
        .to_latex('simd-table.tex', hrules=True)
)
```