MSc in Artificial Intelligence and Data Science

Dr. Marika Asgari

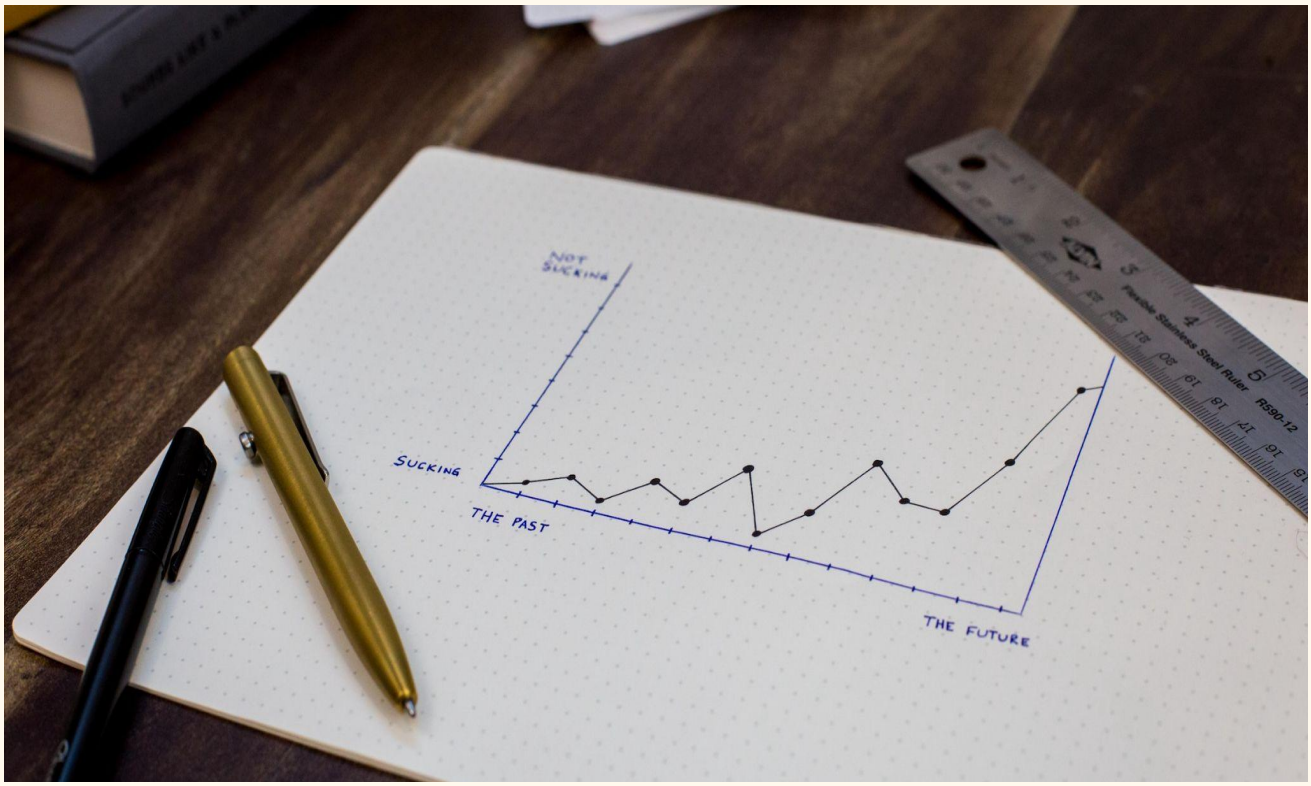

Image 1: Charting Goals, (Isaac Smith, Unsplash).

771766

Fundamentals of Data Science Project

By Daniel Akinbankole

Table of Contents

Introduction

Every ten years, the United Kingdom undertakes a census of the population, with the most recent one having been conducted in 2021. The purpose of such a census is to compare different people across the nation and to provide the government with accurate statistics on the population to enable better planning, develop policies, and allocate certain funding.

The census data contains the following field:

1. Street Number.
2. Street Name.
3. First Name of Occupant.
4. Surname of the occupant.
5. Age of occupant.
6. Relationship to the “Head” of the household (anyone aged over 18 can be a “Head” – they are simply the person who had the responsibility to fill in the census details.)
7. Marital status (one of Single, Married, Divorced, Widowed, or “NA” in the case of minors).
8. Gender (one of: Male, and Female).
9. Occupation.
10. Infirmity.
11. Religion.

Project Aims

The town from the census data is a modestly sized one sandwiched between two much larger cities that are connected by motorways. The town does not have a university, but students do live in the town and commute to the nearby cities.

As part of the local government team, the project aims are to:

- ★ Clean the data in the census data.
- ★ Analyse the cleaned data so as to make a decision on what to do with an unoccupied plot of land and what to invest in the town.

Cleaning the Data

The step taken to clean the census data in order to ensure the data are useful and functional toward the intended analysis are:

- Fixing improper data types.
- Cleaning the empty data entries
- Cleaning the NaN entries

Fixing improper data types

Upon inspecting the Dataframe information using the ***DataFrame.info()*** method. the Age Column data type was set to 'float64'

```
1 census_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9426 entries, 0 to 9425
Data columns (total 11 columns):
 #   Column                          Non-Null Count  Dtype
---  ------                          --------------  -----
 0   House Number                    9426 non-null   int64
 1   Street                          9426 non-null   object
 2   First Name                      9426 non-null   object
 3   Surname                         9426 non-null   object
 4   Age                             9426 non-null   float64
 5   Relationship to Head of House   9426 non-null   object
 6   Marital Status                  6996 non-null   object
 7   Gender                          9426 non-null   object
 8   Occupation                      9426 non-null   object
 9   Infirmity                       9426 non-null   object
 10  Religion                        6937 non-null   object
dtypes: float64(1), int64(1), object(9)
memory usage: 810.2+ KB
```

Image 2: Image showing the information about the census data frame.

The solution was to change the Age column data type from 'float64' to 'int64' by using the ***DataFrame.astype('int64')*** method.

Cleaning the Empty Entries

```
In [8]:  1 empty_entries = census_df == ' '
         2 empty_entries.sum()

Out[8]: House Number                    0
        Street                          0
        First Name                      1
        Surname                         0
        Age                             0
        Relationship to Head of House   0
        Marital Status                  1
        Gender                          0
        Occupation                      1
        Infirmity                       7
        Religion                        1
        dtype: int64
```

Image 3: Image showing the sum of the empty entries in the census data frame .

Upon inspecting the Dataframe for Empty entries the columns containing empty entries are:

- First Name.
- Occupation.
- Marital Status.
- Infirmity.
- Religion.

- **First Name Column**

  The first name column has just one empty entry which was replaced with *"Unknown".*

- **Occupation Column**

  The occupation column has just one empty entry, after further investigating the entry has *"9"* in the Age column, hence the empty entry was replaced with *"Student".*

- **Marital Status Column**

  The marital status column has just one empty entry, after further investigating the entry has *"Husband"*, in the Relationship to the head of the house column, hence the empty entry was replaced with *"Married".*

- **Infirmity Column**

  The infirmity column has seven empty entries which were replaced with *"None".* Because that was the most occurring entry in the column.

- **Religion Column**

  The religion column has just one empty entry which was replaced with *"Unknown".*

Cleaning the NaN Entries

```
In [23]: 1 census_df.isna().any()

Out[23]: House Number                    False
         Street                          False
         First Name                      False
         Surname                         False
         Age                             False
         Relationship to Head of House   False
         Marital Status                   True
         Gender                          False
         Occupation                      False
         Infirmity                       False
         Religion                         True
         dtype: bool

In [24]: 1 census_df.isna().sum()

Out[24]: House Number                       0
         Street                             0
         First Name                         0
         Surname                            0
         Age                                0
         Relationship to Head of House      0
         Marital Status                  2430
         Gender                             0
         Occupation                         0
         Infirmity                          0
         Religion                        2489
         dtype: int64
```

Image 4: Image showing the sum of the NaN entries in the census data frame.

Upon inspecting the data frame using the ***DataFrame.isna()*** method for NaN entries, the columns containing NaN entries are:

- Marital Status.
- Religion.

- **Marital Status Column**

```
In [26]:
# getting the description of the nan entries in the Marital Status
marital_nan.describe()
```

Out[26]:

	House Number	Age
count	2430.000000	2430.000000
mean	52.688477	9.045679
std	55.148675	5.091017
min	1.000000	0.000000
25%	11.000000	5.000000
50%	31.000000	9.000000
75%	78.000000	14.000000
max	230.000000	17.000000

```
In [27]:
# getting the unique values of the nan entries in the Marital Status
marital_nan['Occupation'].unique()
```

```
Out[27]: array(['Child', 'Student'], dtype=object)
```

Image 5: Image showing the description and unique entries of the marital NaN data frame

---

The marital status contains ***2430*** NaN entries, after investigating those entries it shows that those entries have:

1. Minimum age of ***0*** years and maximum age of ***17*** years.
2. Unique occupation entries are ***"Child"*** and ***"Student"***.

Hence, the NaN entries in the Marital column were replaced with

1. *"Single Child"* for those entries with the *"Child"* occupation label.
2. *"Single Student"* for those entries with the *"Student"* occupation label.

- **Religion Column**

```
In [30]:
1 # Getting the unique values in the religion column
2 census_df['Religion'].unique()

Out[30]: array(['None', nan, 'Catholic', 'Methodist', 'Christian', 'Muslim',
       'Private', 'Jewish', 'Sikh', 'Nope', 'Buddist', 'Baptist', 'Sith',
       'Bahai', 'Agnostic', 'Orthodoxy', 'Unknown'], dtype=object)

In [31]:
1 # Getting the series description
2 census_df['Religion'].describe()

Out[31]: count     6937
         unique      16
         top       None
         freq      3114
         Name: Religion, dtype: object
```

Image 6: Image showing the unique entries, and description of the Religion column in the census data frame.

Replaced the nan entries in the Religion column to *"None"* because that is the most occurred value in the column, that is the Mode.

The Project Task

Task 1

What should be built on an unoccupied plot of land that the local government wishes to develop?

- **Task 1 Analysis:** The age pyramid shows the distribution of various age groups for each gender of the census data frame. The population count is shown for each five-year interval starting from 0-4 and continuing up to 100+.

  Dividing this age interval into groups namely:

  - Pre-reproductive Age (0 - 14)
  - Reproductive Age (15 - 44)
  - Post Reproductive Age (45 upwards)

Pre-reproductive (0 - 10 years)

The pre-reproductive age group consists of individuals who have not yet reached the reproductive age (Popedadmin, 2019). This age group 0 to 14 represents the base of the population pyramid.

Reproductive Age (15 - 44 years)

The reproductive age group consists of individuals who have reached the reproductive age, these are commonly referred to as the child-bearing age (Popedadmin, 2019). This age group 15 to 44 represents the middle section of the population pyramid.

Post-reproductive Age (45 year and upward)

The post-reproductive age group consists of individuals who have exceeded the reproductive age and are no longer able to reproduce (Popedadmin, 2019). This age group 45 and upwards represent the top of the population pyramid.

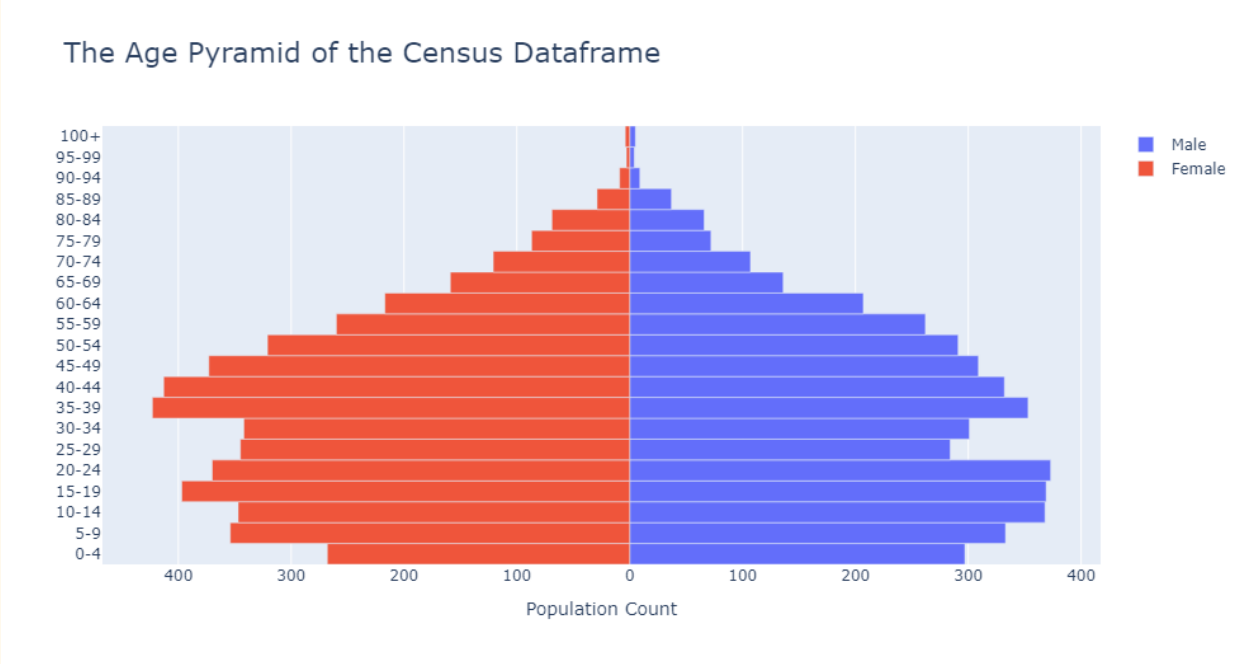

Image 7: Image showing the Age pyramid of the census Dataframe

```
In [49]:
# checking the marital staus of the reproductive age (15-44)
repro_df['Marital Status'].describe()

Out[49]:
count       4302
unique         5
top       Single
freq        2309
Name: Marital Status, dtype: object

In [50]:
# occupation description in the reproductive age population dataframe
repro_df['Occupation'].describe()

Out[50]:
count                   4302
unique                   632
top       University Student
freq                     608
Name: Occupation, dtype: object

In [58]:
# most occured Age in the reproductive age population data frame
repro_df['Age'].mode()

Out[58]:
0    15
Name: Age, dtype: int64
```

Image 8: Image showing the description and mode of the reproductive data frame

---

In the census data, 46% of the population with a count of 4302 individuals of reproductive age out of the total count of 9426 individuals in the town.

The pre-reproductive age (0-14) has the lowest population count with 1967 individuals (approximately 21%) to the total count of 9426 individuals in the town. Because there is less population in the pre-reproductive age group than they are in the reproductive age group, the population will grow more slowly as the number of people reaching their reproductive year decreases.

Task 1 Conclusion:

Further investigation of the reproductive data frame shows that the most occurring entries in the occupation column are University students with the most occurring age of 15 years which is below the legal driving age in the UK (Government Digital Service, 2015b), with there being no university in the town this set of individual will need to commute to the nearby cities. Building a train station on the unoccupied plot of land that the local government wishes to develop will benefit the town.

Task 2

1. What should be Invested in?

Task 2 Analysis: Upon observing the post-reproductive age (45 upward) which counts 3157 (33%) of the total count of 9426 population.

In [152]:
```
# Filtering age 45 upwards, Post Reproductive age
post_repro_df =census_df.loc[(census_df['Age'] >= 45) ]

# describing the reproductive age
post_repro_df.describe()
```

Out[152]:

	House Number	Age
count	3157.00000	3157.000000
mean	46.49414	59.588217
std	52.46467	11.476552
min	1.00000	45.000000
25%	10.00000	50.000000
50%	24.00000	57.000000
75%	60.00000	66.000000
max	229.00000	105.000000

Image 9: Image showing the post reproduction data frame description

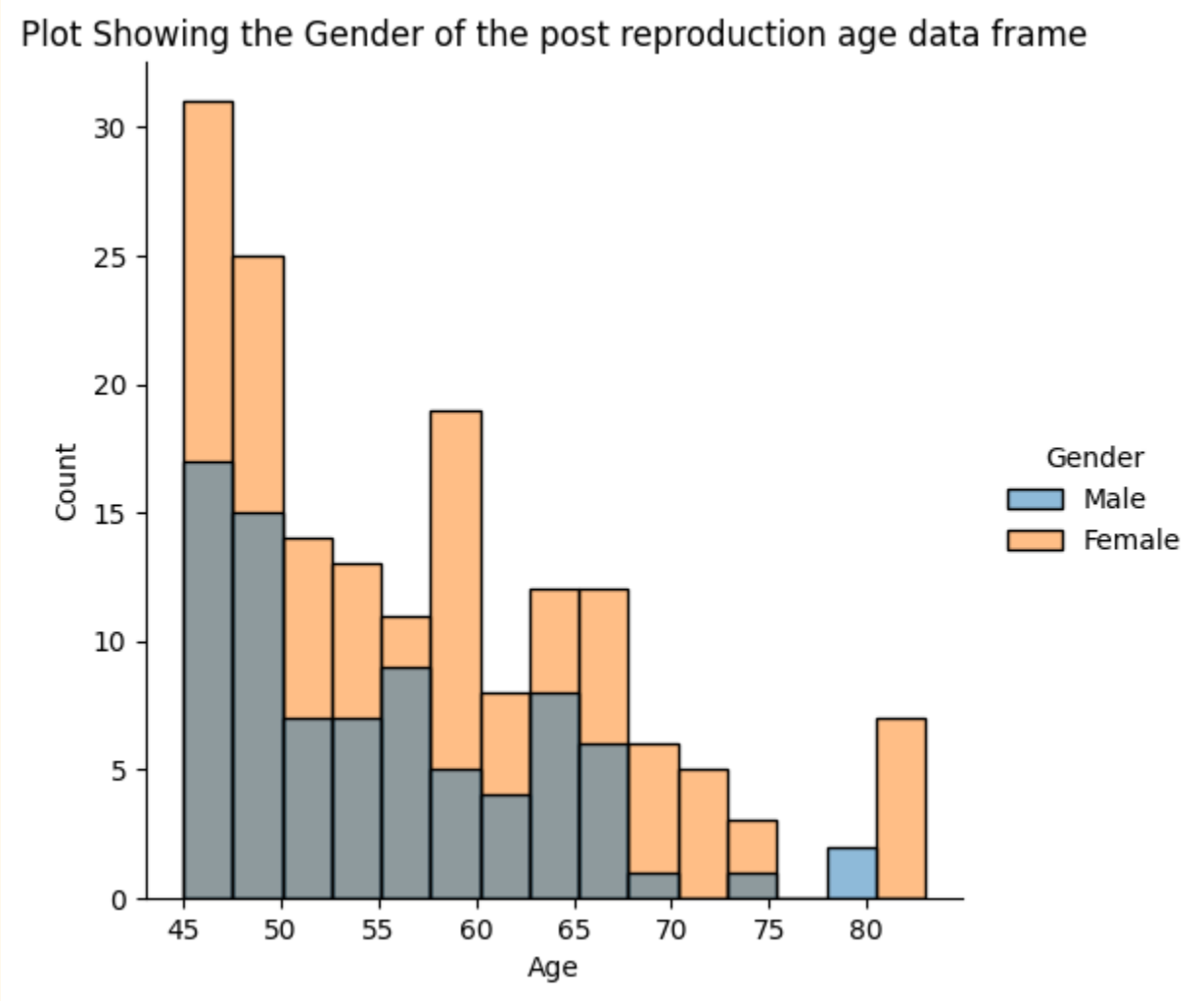

Image 10: Image showing the plot of Gender of the post reproductive age data frame

---

Task 2 Conclusion:

Because there are lots of population in the post-reproductive age group, the town will need to invest and allocate more funds for end life and old age care because there will be an increase in number of retired people in the future year.

Reference List

1. Government Digital Service (2015b) “Driving lessons and learning to drive,” GOV.UK [Preprint]. Available at: https://www.gov.uk/driving-lessons-learning-to-drive#:~:text=You%20can%20start%20driving%20a,you%20can%20learn%20to%20drive.

2. Popedadmin (2019) How Reproductive Age-Groups Impact Age Structure Diagrams | Population Pyramids - Population Education. Available at: https://populationeducation.org/how-reproductive-age-groups-impact-age-structure-diagrams-population-pyramids/.