**Assignment Report for Certificate in Introductory Data Analytics**

GitHub repository URL: <https://github.com/howletts/UCDPA_SarahHowlett>

**Say what each folder has** analyze\_data.py

modified: data/UCDPA\_SarahHowlett\_report.docx

deleted: data/surf\_venue\_count\_per\_region.png

modified: visualize\_data.py

Develop a Python project to analyse real world scenarios and generate valuable insights by visualizing information. The project aims to analyse data from different data sources, manipulate information and visualize to generate insights.

A student can use any open-source dataset available online for analytics.

● Describing your process, dataset, different sources, graphs and insights.

● Justify the use of each learning outcome concept for eg.. Why did you use list over dictionary?

**1** Project should use a real world dataset and include a reference of their source in the report

I downloaded "Attractions.csv" from https://data.gov.ie/dataset/attractions and saved it locally to demonstrate

reading a csv file into a dataframe.

For the attivities and accommodation datasets i made a call to the APIs https://failteireland.azure-api.net/opendata-api/v1/accommodation

https://failteireland.azure-api.net/opendata-api/v1/activities

Using SQLite Developer I created a demo database "sqlliteDB\_ucdproj.db with a table COUNTY\_PROVINCE\_LINK to demonstrate importing data from a Relational Database - see "data" folder.

I used python pickles to store the data temporarily to avoid constantly hitting the endpoints during development.

**2** Importing data

• Your project should make use of one or more of the following, [1]

• Relational Database or API or Web Scraping

• Import a CSV file into a Pandas DataFrame. [1]

**Analyzing data/Python:**

1. Analyse and process the activities dataset  
  
The API call was filtered to search for records tagged with all three surfing activities; surfing, windsurfing and kitesurfing.   
  
The data was processed as follows:  
The 'address\_region' (county) was extracted from the 'address' column using a **reusable custom function** and this was added as new column to the dataframe as it is needed for joining.  
The data was **sorted** by 'name' and 'address\_region' (county) in ascending order to prepare for removing duplicates.  
A small number of records had duplicate 'name' and 'address\_region', they only differed on the tag field, **duplicates were removed** and the last record was kept because it seemed to have more tags in general.   
The mains fields of interest were extracted using **.loc** to select 'name', 'address\_region', 'tags' columns.  
The number of rows **grouped by** 'address\_region' (county) were counted and saved in a new DataFrame df\_surf\_venue\_per\_region for analysis.

The df\_region\_type DataFrame is a lookup table for counties ('address\_region' ) which contain a city. df\_surf\_venue\_per\_region and df\_region\_type were **merged** using a **left join** because all rows from df\_surf\_venue\_per\_region are needed. The columns to join on are specified as they have different names in both DataFrames.   
The majority of counties have no cities so missing values were **replaced** with 'No City' using **fillna**.

2. Analyse and process the attractions dataset

In this case there was no filtering applied to the attractions dataset on retrieval as the entire csv was read into a DataFrame.

The data was processed as follows (similar to above):

**Duplicate** records were dropped.  
As small number of records that didn’t have an 'AddressRegion' field populated were **dropped** as this field is needed for analysis/joining.  
Other empty cells in the other columns were filled with 'Unknown' using **fillna**.  
Only columns of interest selected and then the DataFrame was **sorted** in ascending order.  
**Group by** 'AddressRegion' and get the count per County ('AddressRegion'), has **index** as default so keep it for joining.  
The df\_county\_in\_province DataFrame is a lookup table for counties ('address\_region' ) to get which province it is in. Set the **index** to 'county' on df\_county\_in\_province for joining efficiency. **Left join** but using **indexes** rather than columns. However, an inner join should work here too, as every county is in a province.

4) Python

• Define a custom function to create reusable code. [1]

make\_api\_call(), download\_csv(), extract\_region, get\_lollipop\_colours

• Numpy. [1]

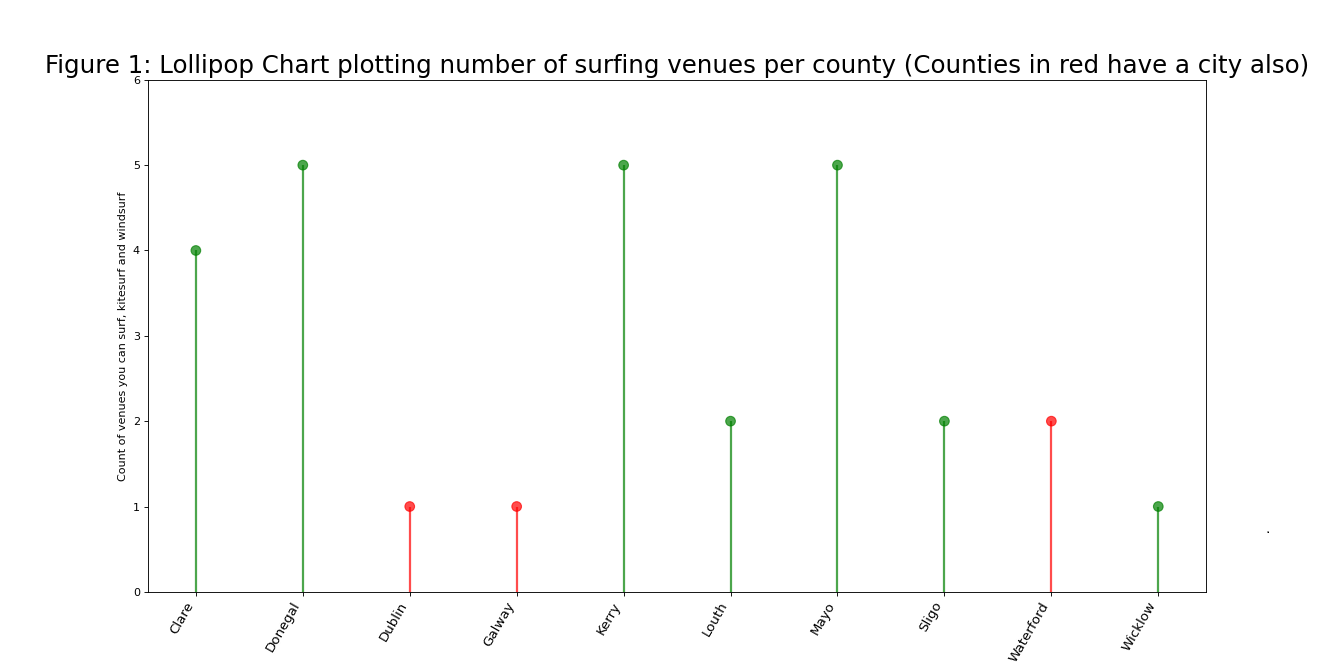
• Dictionary or Lists. [1] –

A list comprehension to make a list of colours

**Visualizations/Insights**

The following is a lollipop chart:

shows the total number of venues which cater for multiple types of surfing within a county in Ireland. The lollipops in red show counties that contain a city, those in green do not. The colour is set based on a DataFrame column, the function to do this uses a **list comprehension to iterate** over the values in the column.

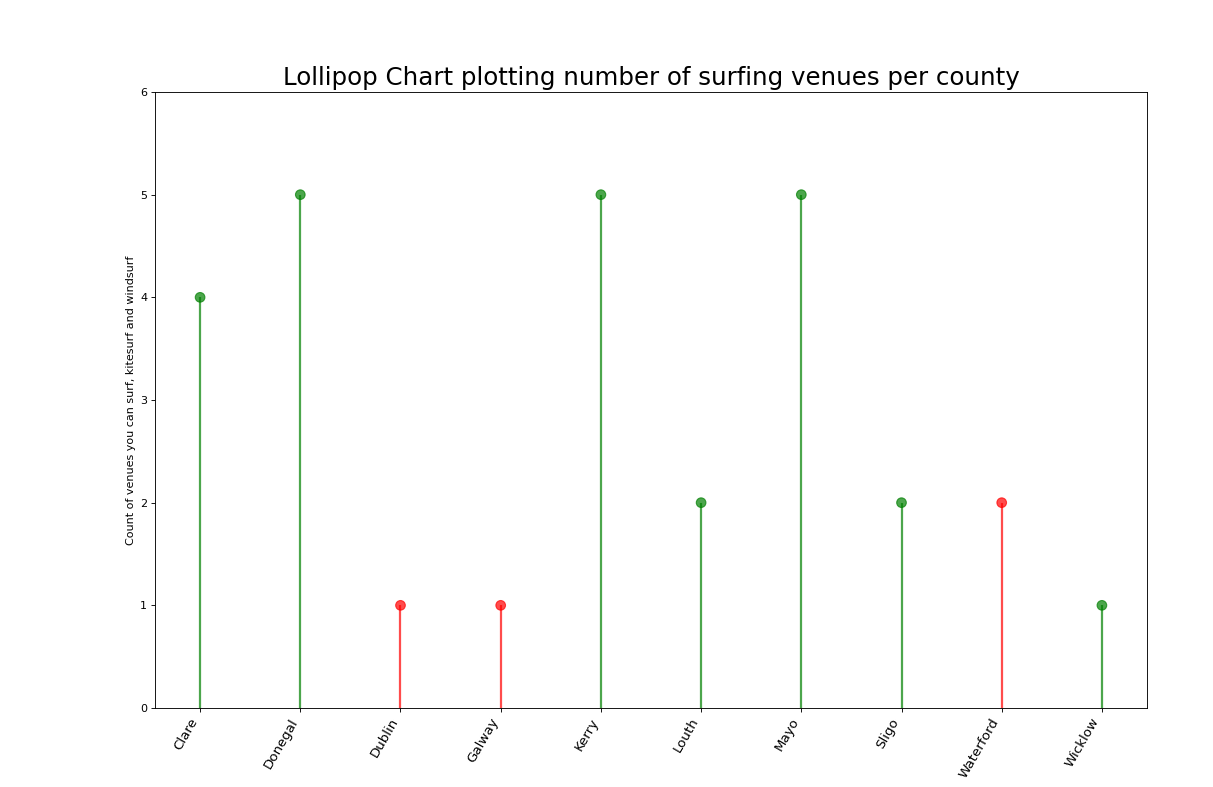
Insights:

* Donegal, Kerry and Mayo have the most venues to choose from.
* Counties with cities have very little options for surfing.

6) Generate Valuable Insights

• 5 insights from the visualization. [2]

**Visualization and Insights:**



**Insight 1:** If you are interested only in locations where you can surf, kitesurf and windsurf the counties with the most options are Donegal, Kerry and Mayo (5 locations each).  **Insight 2:** Counties that contain a city do not have much options (see red lollipops).