Project Report

1. GitHub URL  
   <https://github.com/dutchdave1708?tab=repositories>

Account includes my notes from lessons as well as the project. The project files are in folder **ucd\_datacamp\_lessons** 🡪 **Master-repository** 🡪 folder **UCD\_Dave\_Project**

Full url: https://github.com/dutchdave1708/ucd\_datacamp\_lessons/tree/master/UCD\_Dave\_Project

1. Abstract(Short overview of the entire project and features)
   1. **Purpose of my project**:   
      to evidence learning across range of data science elements, including Python (data loading, cleaning, filtering, custom function, regular expression, etc to load/process/use data for insights), Data Visualisation, Machine Learning and describing insights.
   2. **Addition**: (17 Nov) – now that I am comfortable with most Python basics and processing/visualising data, I am going to explore some more machine learning scenarios with logic you come across everyday, like ‘recommend a movie based on another movie’ : vector analysis on text
   3. **Addtion (22 Nov)**; adding unsupervised learning to it, by doing clustering on a wine dataset, producing various charts along the way.

**2.2 Description:**   
I’m using a couple of different data sets to evidence (to myself and you) the learning, leveraging copying a csv, retrieve csv from url, webscraping, MySQL database and API.   
For each section I will decide which dataset is best to use, as my main objective is to evidence learning and put the practise into action. As such, there is no intention at the outset to have an end-to-end project with a particular purpose throughout.

**2.3 Key features**

The key features are:

* Relevant, well commented code ( but note, I have paid no attention to it being efficient or optimised)
* Showcase an understanding of the topics
* A few charts to complement the findings / code / logic
* Data from different sources / different methods to access
* A range of actions on the data, not necessarily because it is useful, but to try different things and work my way through errors.
* You will find the code include:
  + Data loading: csv , url, SQL, API (pending),
  + Filtering dataframes on columnname, column values, reg expressions
  + Data cleaning: empty values, outliers, unused columns, conversion (e.g milligram to gram) etc.
  + Dataframe manipulation: slicing, filtering
  + Merging of dataframes on columnvalues, updating columnname, etc
  + Using numpy, dictionaries & lists to cater for specific actions
  + Charting with matplotlib and seaborn
  + Custom functions
  + Iterators
  + Regular expressions
  + Correlation calculations, linear-regression, R^2 and K-CV calculations
* **Addition 17 Nov:** Machine learning scripts to try out different logic, scikit-learn.
* **Addition 22 Nov:** KMeans , histogram of all measures, cluster centre visualisation, measure & chart inertia,
* **Addition 22 Nov**: added a Dendrogram on the recipe dataset

Introduction  
I used datasets on Recipes, Lego, Movies as it allowed me to evidence learning whilst using a dataset I can relate to. As you will see in the sequency of scripts, I started simple and worked my way through the data to get to something useful and – mainly – try all aspects of Python / Data Science learned over the last 3 months. As the work has progressed, I got a lot more comfortable with python as well as how to quickly find the mistakes, etc and tackled more complex things.

Datasets  
The main dataset is one for Recipes, downloaded from Kaggle : epi\_r.csv. {https://www.kaggle.com/hugodarwood/epirecipes?select=epi\_r.csv}

The file contains ~20000 recipes, where the ingredients have been flipped to columns and the rows are float values to represent either Y/N (1/0) or the amount of ingredient or nutrition value. Except the title column which is string.

I used this file because a) it has a good number of rows, b) it is data that required filtering and cleaning, c) it is ‘real life’ data I understand and d) it allows from some scenarios to work through and chart.

Note: to proof other parts of learning I have also (1) accessed rebrickable csv via url, which is a lego dataset and (2) API for rebrickable (3) created an online MySQL database and write/read via Python

**Addition 17-Nov**: also added IMDB movie data set for machine learning scripts to make recommendations based on movie descriptions.

**Addition 22-Nov**: wine.csv a dataset with numeric values of many characteristics of wine

Implementation Process (Describe your entire process in detail)

Reality is that I started without a plan, but with the basics of ‘how do I get a dataset into a dataframe?’ and go from there.   
The filestructure in the project reflects the evolution of that:

1. Part1\_1\_Load\_EploreData.py: “getting to know the data”
   1. loading the csv into dataframe to then do various exploring of the data to understand the structure, the values.
   2. .info(), .describe(), .head(), a ‘for’ loop to print all 680 columns headers,
   3. remove duplicate titles with .drop\_duplicates
   4. some filtering using .loc with OR ( | ) and AND ( & ) statement
2. Part1\_2\_Selection.py: “process the data to clean it”
   1. Drop duplicates
   2. Drop drinks recipes
   3. Select desired columns only into new df
   4. Drop rows with empty values ( can be done with .dropna but I did column by column as originally I had various interim steps)
   5. Create some scatter charts to explore outliers ( can be done with describe and other methods, but wanted to chart something)
   6. Use .mean and .max / .min to establish sense of quality
   7. Remove values that are x-standard deviation from mean
   8. Chart again to see difference
   9. See how rating and other values have changed pre- and post clean up
3. Part1\_3\_CorrelationMap.py: “explore if there is correlation between values”+charts
   1. Same cleanup as done in 2, but without all the prints/charts/comments
   2. 2 correlation methods Pearsons and Kendall
   3. Create Seaborn heatmaps with labels
4. Part1\_3\_2 Custom Function:
   1. Create custom function to clean up dataframe
   2. Incorporate iterator in the logic
   3. Use lists as input parameters as well
5. Part1\_4\_Regression.py: “apply regression learning”
   1. Import same file, but keep all columns
   2. Dropna to drop all empty values
   3. Remove title column as this is type string
   4. Remove outliers (2 rounds) +change Sodium from milligram to grams
   5. I’ve done different features & targets scenarios
      1. Produce scatter chart on 1 feature and target
      2. Fit regression model (LinearRegression)
      3. Prediction via reg.predict feeding a Numpy array
   6. Using test & train data
      1. 70-30 split, randomstate = 42
      2. Fit model
      3. Calculate R^2 value (outcome = 0.63)
      4. Calculate Root mean^2 (outcome = 118)
      5. Do k-fold CV (cv = 5) [0.627 0.60 0.64 0.64 0.61]
6. Part2\_1\_url\_csv: “load csv from url, and merge dataframes”
   1. Load 2 fiels rom rebrickable.com website
   2. Read the csv’s into a dataframe
   3. Merge dataframe on a columns
   4. Tidy up columnnames
7. Part2\_2\_API. [on hold]
   1. To finish, API key didn’t generate. I haven’t followed up yet as I’m comfortable with the logic and wanted to spend time on ML.
8. Part2\_3\_Webscrape & regular expressions
   1. Load Wikipedia page
   2. Use BeautifulSoup to extract text from html & extract title
   3. Use regex to
      1. Findall
      2. Replace & Sub
      3. Find URLs in text with complex regex.   
         Explanation in comments in script.
9. Part2\_4\_Database
   1. Create via freesqldatabase.com
   2. Create table for phpMyAdmin.co
   3. In python:
      1. Connect to databaseserver
      2. Extract data from table
      3. Create new table
      4. Insert data into new table
      5. SQL Join statement to extract data
10. Part 3\_1\_Machinelearning: ML, charting, custom functions, ..
    1. Load movie data, merge dataframes
    2. Find different types of movie / various values
    3. Vote distribution chart
    4. Calculate Weighted Votes via custom function
    5. Charts on : best by weighted\_vote, most popular, best by unweighted\_vote, highest revenue
    6. ML:
       1. TfidVectorizer
       2. Create Cosine similarity matrix
       3. Function to find similar movies
11. Part 3\_2\_ML2\_Clustering: Kmeas, intertia, custom functions, charting…
    1. Load data
    2. Histogram of all features
    3. Clustering ( via variable, tried different number of clusters)
    4. Find centre of clusters / include on the charts
    5. Measure inertia, charts inertia per # of clusters,
12. Part\_3\_3\_Dendrogram
    1. Create Dendrogram from recipe data
    2. Only on top 50 because 10000+ was too many

Results – 1 – Recipes data   
Results are mostly commented / printed in the scripts, for example the results of data processing or dataloading. Some findings summarised here

* ’42 recipes are Christmas Eve recipes but not Christmas recipes’…  
  it does get a bit better than that:
* Mean for Calories, before removing outliers: 7186.7 after: 426.8
* Mean for Sodium, before removing outliers: 7.1 after: 0.53 (converted to milligrams)
* Mean for Fat, before removing outliers: 393.54 after: 23.8
* Mean for Protein, before removing outliers: 99.7 after: 17.6
* 🡪 the values seem to be in a decent normal range as expected
* How rating has evolved Original: 3.71 Unique: 3.72 No Nulls: 3.84 NoOutliers: 3.83
* Charts checking for outliers (scatter charts using matplotlib)  
  Chart, scatter chart

  Description automatically generatedChart

  Description automatically generated
* Checking correlations, using Kendall and Pearsons
* Charts using Seaborn

Chart

Description automatically generatedChart

Description automatically generated

* Fitting LinearRegression model, train/test data

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

Results 2 – Movie data

* Differences in top 6 movies based on ‘Popularity’, ‘Revenue’, ‘Vote-average’ and (my calculation) ‘Weighted average vote’ (see charts)
* Similar movies to ‘Spectre’ are :   
  Text

  Description automatically generated

Charts – see next page

Graphical user interface, application, table

Description automatically generatedGraphical user interface, application, table

Description automatically generatedGraphical user interface, application, table

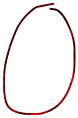
Description automatically generatedChart, bar chart

Description automatically generatedChart, funnel chart

Description automatically generatedChart, histogram

Description automatically generatedChart, histogram

Description automatically generated



Results Wine

3.1 Histograms of all features of data set. (Customer\_segment to be removed)\_

Graphical user interface, application

Description automatically generated

3.2 Scatter of wines on 2 features, including cluster centre

Chart, scatter chart

Description automatically generated

3.3 Inertia measure, per 1-10 clusters. (3 or 4 clusters is best)

Chart, line chart

Description automatically generated

Results 4: Dendrogram Recipe data

Chart, histogram

Description automatically generated

Insight

* Recipe 1- Surprisingly *No* correlation between calories or other values and Rating
* Recipe 2 - Expected correlation between fat and calories
* Recipe 3 - Unexpected: correlation between Sodium and Protein
* Movies 1 – The movies with most revenue do not appear on the “best movies’ top
* Movies 2 - Best movies weighted votes different to ‘normal vote’ score
* Movies 2 – Recommendations based on movie descriptions seem good(!)
* Movies 3 – The ones with low number of votes and low rates, got biggest adjustment in Weighted Rating. (still rubbish movies)  
  Text

  Description automatically generated
* Movies 4 – whilst those with high scores & high vote count have minimal change
* Text

  Description automatically generated
* Wine1 – Clustering in 3 groups is best fit for that dataset

References  
Mainly used

* Kaggle to find different datasets to practise on. Link to dataset used in main part is above
* Stackoverflow for many queries on the syntax
* Regex101 with help on building a more complex regular expression
* Realpython.com; articles on custom definition creation
* Towardsdatascience.com: various python ‘how to’ queries
* Datacamp for many tutorials