CS 210
Project 7 – Data Analysis

1. Introduction

This project allows you to hone your skills in working with dictionaries and reading data from CSV files. You will experiment with a common benchmark from Machine Learning, the Titanic Dataset.

The files `titanic-clean.csv` and `titanic.py` provide the dataset and a template for your solution. You can find those files on Canvas and Coding Rooms.

1.1. Project goals

This week's project will provide an arena where you can:

- Practice working with complex dictionaries.
- Continue practicing CSV file input and matplotlib visualization.
- Write structured code.

All the functions corresponding to different parts of this project will be in the same Python module, `titanic.py`, but Coding Rooms will test and grade them individually. The template file provides a sample `main()` function designed to test the functions you must write. You are responsible for writing those functions.

1.1. How to Succeed

Remember the following tips:

- Start early.
- Read, understand, and devise a solution before starting coding.
- You may consult your notes, slides, and exercises posted on Canvas or perform Google searches.
- You may copy one line or two of code, but not complete solutions.

2. Data

In this project, you will work with a cleaned-up version of the Titanic Dataset, `titanic-clean.csv`, which contains data on the passengers of the Titanic, which sank infamously in 1912 after striking an iceberg (https://tinyurl.com/mu3jdjcc). The following table provides brief descriptions of the fields in this dataset:

Field name	Type	Description
PassengerID	int	Unique ID for each passenger
Survived	int	1 if the passenger survived, 0 if the passenger died
Pclass	int	The class in which the passenger was traveling (1 - best, 2, or 3 -worst)
Sex	str	'male' or 'female'
Age	float	Passenger age
SibSp	int	The number of siblings or spouse of a person on board
Parch	int	Similar to the SibSp, this feature contains the number of parents or children
Fare	float	The fare paid by the passenger
Embarked	str	The port from which the passenger embarked
FamilySize	int	A combination of SibSp and Parch
age_group	str	Derived from age – the age group for this passenger, e.g., '20-29'

The template code directory in Coding Rooms and Canvas contains the **`titanic-clean.csv`** dataset. It is a good idea to first look through **`main()`** and ensure you understand the meaning and use of the functions you will write.

3. Project Activities

Solve the following problems and submit your work on Coding Rooms

3.1. [20 pts] Load the Titanic data from a CSV file to a dictionary

Requirements:

- Define the function **`load_data(file_name:str, types:dict)->dict`**, which opens and reads the file named **`file_name`** and creates a **`dictionary`** containing the data described in **`types`**.
- The function **`load_data`** creates a dictionary with the same structure as **`types`**, containing the data corresponding column as a list. For example, the column **`'Survived'`** in the CSV file is now stored in the dictionary under the key **`'Survived, '`** which stores a list (the column data).
- You may use a **`csv.reader`** to read the file – recall that the first line of the file contains the column names.
- The returned dictionary only contains the data in the columns specified in **`types`**.

Note to Windows users: You may have to add an extra argument when you call **`open`**, e.g., **`open(filename_str, 'r',encoding='utf8')`**.

Example:

The following set of instructions

```
>>> titanic_types = {'PassengerId': int,
                     'Survived': int,
                     'Pclass': int,
                     'Sex': str,
                     'Age': float,
                     'SibSp': int,
                     'Parch': int,
                     'Fare': float,
                     'Embarked': str,
                     'FamilySize': int,
                     'age_group': str}
>>>data = load_data('Titanic-clean.csv', titanic_types)
>>>for key, val in data.items():
    print(key, val[:4])
```

would output the data for the first 4 passengers in the dataset:

```
('PassengerId', <class 'int'>) [1, 2, 3, 4]
('Survived', <class 'int'>) [0, 1, 1, 1]
('Pclass', <class 'int'>) [3, 1, 3, 1]
('Sex', <class 'str'>) ['male', 'female', 'female', 'female']
('Age', <class 'float'>) [22.0, 38.0, 26.0, 35.0]
('SibSp', <class 'int'>) [1, 1, 0, 1]
('Parch', <class 'int'>) [0, 0, 0, 0]
('Fare', <class 'float'>) [7.25, 71.2833, 7.925, 53.1]
('Embarked', <class 'str'>) ['Southampton', 'Cherbourg', 'Southampton',
'Southampton']
('FamilySize', <class 'int'>) [1, 1, 0, 1]
('age_group', <class 'str'>) ['20-29', '30-39', '20-29', '30-39']
```

3.2. [20 pts] Data summaries

Requirements:

- Define the function **`summarize(data:dict)`** that takes the dictionary returned by **`load_data`** as the **`data`** parameter and summarizes each dictionary value, printing the results and right-aligning the numerical results for better readability.
- For **`int`** and **`float`** values, calculate the min, max, mean, standard deviation, and mode (you may use built-in functions and the statistics package).
- For non-numerical data (e.g., strings), output the number of unique values (hint: you may use a Python **`set`** to eliminate repeated values) and the most common value.

Example output:

```
>>> summarize(data)
Statistics for PassengerId:
    min:    1.0
    max:  891.0
   mean:  446.0
  stdev:  257.4
   mode:    1.0
Statistics for Survived:
    min:    0.0
    max:    1.0
   mean:    0.4
  stdev:    0.5
   mode:    0.0
Statistics for Pclass:
    min:    1.0
    max:    3.0
   mean:    2.3
  stdev:    0.8
   mode:    3.0
Statistics for Sex:
Number of unique values: 2
      Most common value: male
Statistics for Age:
    min:    0.4
    max:   80.0
   mean:   29.3
  stdev:   13.3
   mode:   26.5
Statistics for SibSp:
    min:    0.0
    max:    8.0
   mean:    0.5
  stdev:    1.1
   mode:    0.0
Statistics for Parch:
    min:    0.0
    max:    6.0
   mean:    0.4
  stdev:    0.8
   mode:    0.0
Statistics for Fare:
    min:    0.0
    max:  512.3
   mean:   32.2
  stdev:   49.7
   mode:    8.1
Statistics for Embarked:
Number of unique values: 3
      Most common value: Southampton
Statistics for FamilySize:
    min:    0.0
    max:   10.0
   mean:    0.9
  stdev:    1.6
   mode:    0.0
Statistics for age_group:
Number of unique values: 9
      Most common value: 20-29
```

3.3. [30 pts] Data Correlations

In this part, you will implement the Pearson correlation coefficient computation described in Section 5.4 in the textbook.

Requirements:

- Define the function **`pearson_corr(x:list,y:list)->float`** that takes two lists of numerical values (**`int`**s or **`float`**s) and returns the Pearson correlation coefficient **`r`** (https://tinyurl.com/kyr64nfy) that the two lists of numbers exhibit. The results in **`r`** lie in the range **`(-1, 1)`**.
- The **`x`** and **`y`** parameters are lists of numerical values (**`int`**s or **`float`**s). Make sure they have the same length. Raise an exception otherwise.
- Round the result to 2 digits after the decimal point.

Example output:

```
Correlation between age and survival is -0.07
Correlation between fare and survival is 0.26
Correlation between family size and survival is 0.02
```

3.4. [30 pts] Visualize Data Relationships

Prerequisite:

You need the **`matplotlib`** Python package (available on Coding Rooms – no installation required). To install **`matplotlib`** on your computer, copy/paste this command: **`pip3 install matplotlib`** in a Terminal. After installation, import it with **`import matplotlib.pyplot as plt`**.

Requirements:

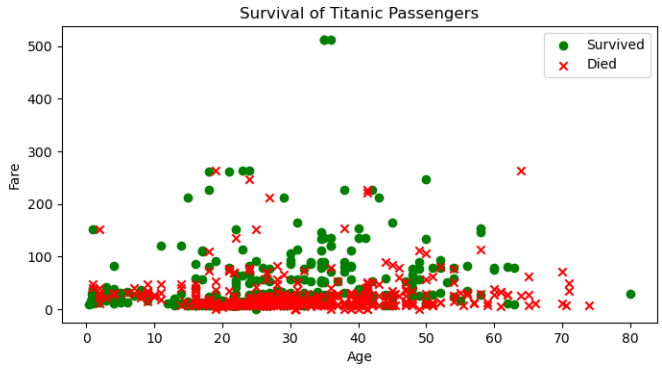
- Define the function **`survivor_vis(data:dict,col_1:tuple,col_2:tuple)->plt.Figure`** to visualize survival in the Titanic dataset as a scatterplot. The **`data`** parameter is the dictionary you created in Section 3.1. The parameters **`col_1`** and **`col_2`** are tuples that contain key values that indicate the data columns to be visualized, e.g., **`('age',float)`** and **`('family_size', int)`**.
- Plot the data for survivors using a different color or marker style than the data for non-survivors.
- Make sure to add **`x-axis`** and **`y-axis labels`** (**`plt.xlabel()`**, **`plt.ylabel()`** with appropriate arguments), and a legend (**`plt.legend()`**).
- Save the plot in a png file named **`f'scatter_{col1[0]}_{col2[0]}.png'`**. Make sure to call the **`plt.savefig`** method *before* you call **`plt.show(block=False)`** to display the plot.

Note: You can control figure dimensions when creating a new figure with the `figsize` parameter, e.g., `figure = plt.figure(figsize=(8,4))`. Also, here is an example call to scatter that enables you to use markers, colors, and a legend (the label argument is what shows up in the legend):
`plt.scatter(x,y,marker='o',c='pink',label='Survived')`.

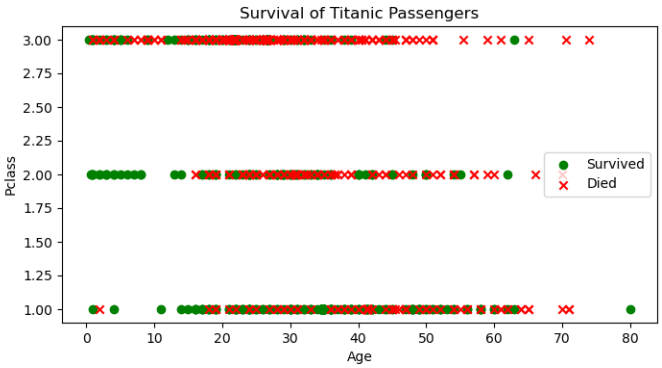
Refer to the `matplotlib.pyplot.scatter` documentation for more details. The function `plt.legend()` generates the legend when you call it without any arguments (after making all the `plt.scatter` calls.)

Examples:

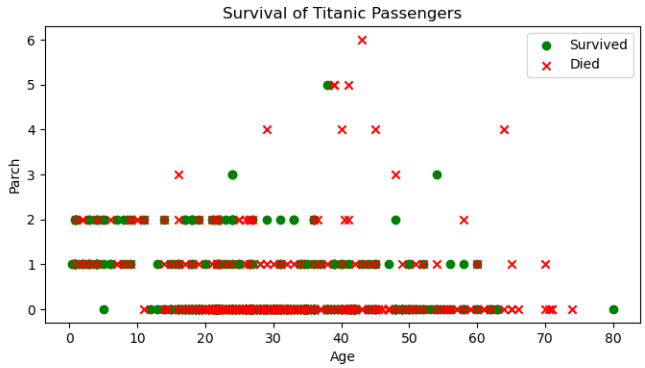
- `scatter_age_fare.png`: Survival for Age vs. Fare – green circles and red x mark survivors and non-survivors, respectively. The plot shows relatively few low-fare passengers survived compared to their wealthier counterparts.



- `scatter-age-pclass.png`: Survival for `age` vs. `pclass` (passenger class) green circles and red x denote survivors and non-survivors, respectively. Since Pclass has only three values, the scatterplot shows three lines – the top indicates the third-class (or poorest passengers), and the bottom is the first-class (or richer) passengers.

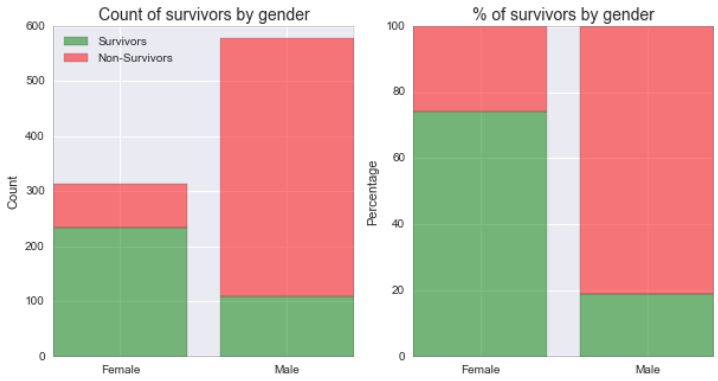


- `scatter-age-parch.png`: Titanic dataset: `survival` for `age` vs. `parch` – green circles and red x denote survivors and non-survivors, respectively. Members of larger family sizes were less likely to survive. You can also plot `family_size` instead of or in addition to `parch` (adding more plotting calls to `main` is fine).



3.5. [10 xc pts] Extra Credit - Survival Odds

In a new file, **`survivors.py`**, write a program to compute and plot (as a bar chart - https://tinyurl.com/4dx8bmf6) the number of survivors split by different variable values (you may choose any variable). Below is an example showing survivors by gender. The left plot shows total counts, while the right one shows percentages of the total. You can import and use your functions from **`titanic.py`**.



4. Submission Summary

Upload the following files to Coding Rooms:

- **`titanic.py`**
- **`survivors.py`**
- **screenshots of your plots if you decide to solve the extra credit problem**

Do not forget to submit your work.

Note: This assignment will be updated with clarifications based on your questions.