Case Study 4

Moral Machine Part 1

Due: XXX

**Moral Machine Part 1**

Welcome to your first day on the job at a fancy new self-driving car company: Messla. At Messla, we represent the cutting edge in self-driving technology, which is why we need to innovate faster than anyone else out there. In order to get our new self-driving cars on the road by the end of this year, we need to come up with the best algorithm to tackle one problem that has gotten in the way of all of our self-driving competitors: **The Trolley Problem**

For your first task on the job, I'm going to need you to create the algorithm that determines *what* our self-driving cars should do in the event of an unavoidable collision. We call our algorithm **The Moral Machine**.

This will take 3 steps:

1. Use the data we have from previous unavoidable collisions to create your model

2. Decide the criteria that you want our cars to use to determine who to kill and who to save

3. Test your model on our data and see how well it works!

Should be easy, right?

**Problem 1 (25 points) Read in our internal information file**

This file represents all of the people that have been a part of an unavoidable collision with one of our autonomous vehicles. Here is how you can interpret this file:

Every even row represents the demographics of the pedestrian, usually someone crossing the street. Every odd row represents the demographics of the driver.

This means that every two rows of this csv represent one accident.

In some of these accidents, the driver was at fault. In some of these accidents, the pedestrian was at fault. This is indicated in the 5th column of every data entry.

The data is formatted as follows:

RowNumber, Age, Gender, Country, Fault

- Where \*RowNumber\* is the number row, even numbers are the pedestrian, odd numbers are the driver

- Where \*Age\* is the age of the driver or pedestrian

- Where \*Gender\* is the gender of the driver or pedestrian

- Where \*Country\* is the country of origin of the driver or pedestrian

- Where \*Fault\* is whose fault the collision was, the driver or the pedestrian

Please read in the csv file titled "*trolley-problem.csv*", skip the header (but print the header), and then print out each line in the csv file as a list. Don't forget to close the file after you finish.

**Part 1:** Complete the task using the csv.reader method()

**Part 2:** Complete the task using the .read() .splitlines() and .split() methods

**Part 3:** Complete the task using the .readline() and .split() method

**Problem 2** **(25 points)** **Parse and clean the file**

You may have noticed that some of this data is numeric, but it is being saved as a string. We need that data as an integer so that we can make comparisons between age and between row number in order to complete our algorithm. We need you to create a list of lists.

Each item in this list of lists is:

- a list that represents a single row in the csv

- each number in the list is an integer datatype

- everything else is a string

Write a function that takes in any file name and returns the list of lists specified above.

Use the following function header:

**def parseAndClean(***fileName***):**

*Hint: It will be easier if you start with an empty listOfLists and use the .append() function!*

**Problem 3 (25 points) Create Criteria For Your Trolley Problem Algorithm**

Take a few minutes to think about what criteria you would use to determine which of the two people in each accident should be saved and which of the two people in each accident should be killed. Write down your thoughts and your criteria as markdown. You should select at least 4 pieces of criteria that are relevant to the data that you have access to. Consider what your edge cases are, what would your decision tree look like?

**Problem 4 (25 points) Code Your Trolley Problem Algorithm**

Now that you know the criteria that you have set for your trolley problem algorithm, it's time to code it! Create a function that takes in two parameters: pedestrian and driver - the two people involved in a specific unavoidable self-driving car collision,

Where *pedestrian* is a list of age, gender, country, and fault

Where *driver* is a list of age, gender, country, and fault

Your function should **return** (based on your criteria) who the car should save as a string. If the pedestrian is saved, your code should return "Pedestrian". If the driver is saved, your code should return "Driver".

Use the following function header:

**def whoDoesTheCarSave(***pedestrian,driver***):**