KENNESAW STATE UNIVERSITY
COLLEGE OF COMPUTING AND SOFTWARE ENGINEERING
School of Data Science and Analytics

Data Science & Classes

Tiana Chargin

Supervised by Dr. Robert Vanderheyden, Ph.D

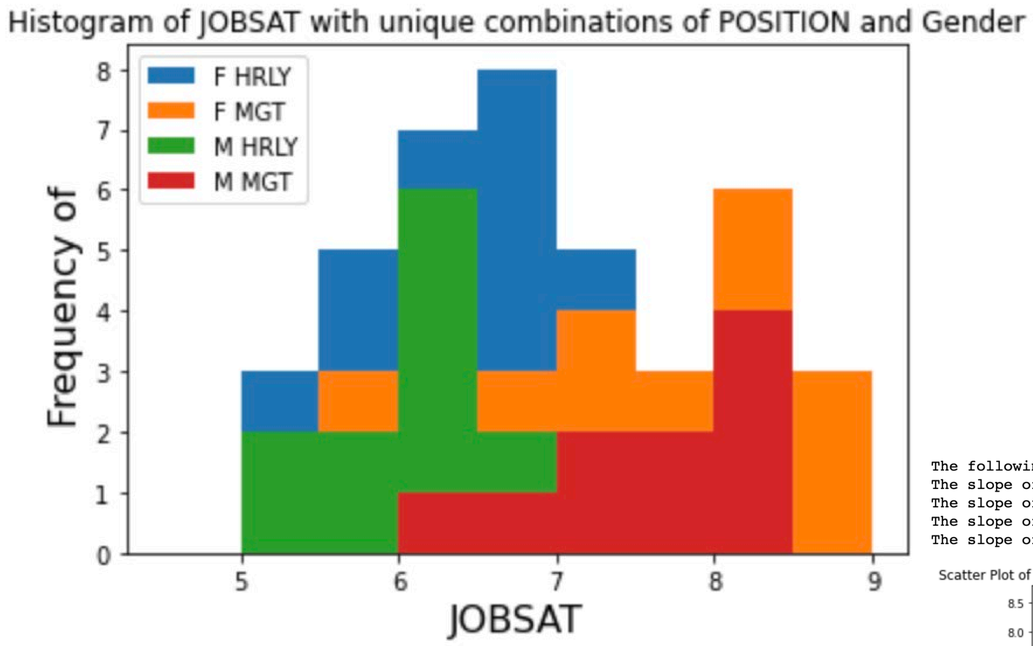
Department of Data Science & Analytics

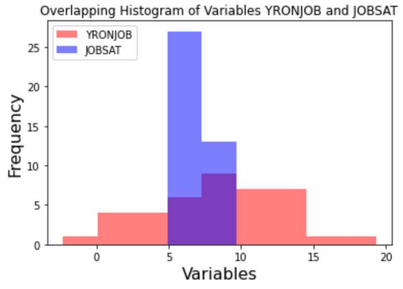
https://github.com/tianachargin

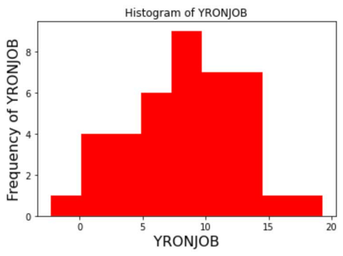
https://www.linkedin.com/in/tianachargin/

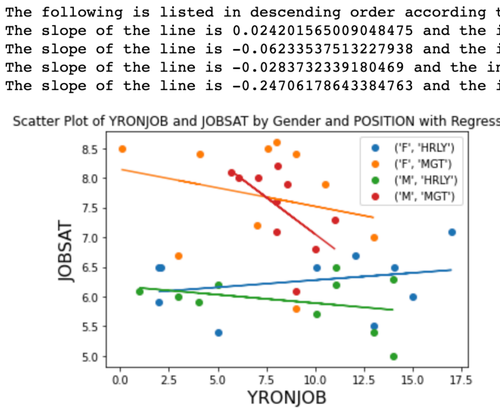
Sample Code

```
class Graphs:
#Constructor with parameters
def __init__(self, quantVar1, quantVar2, qualVar1, qualVar2):
        self.A = quantVar1   #First quantitative variable
        self.B = quantVar2   #Second quantitative variable
        self.C = qualVar1    #First qualitative variable
        self.D = qualVar2    #Second qualitative variable

#Function that calculates bin width for the histogram
    def bin_width(self):
        #Import libaray
        import math
        #Create variable to create array for bins
        #Find min of column
        min = data[self.A].min()
        #Find max of column
        max = data[self.A].max()
        #Find the the count of rows (number of data/size/n)
        index = data.index
        number_of_rows = len(index)
        #Calculate number of bins and round up
        num_of_bins = (math.ceil(math.sqrt(number_of_rows)))
        #Calculate bin width (max - min)/# of bins
        bin_size = ((max - min)/num_of_bins)
        #Round bin width to one decimal place
        increment_bin = round(bin_size, 1)
        #Start bin
        start_bin = (min - increment_bin)
        #End bin
        end_bin = (max + increment_bin)
        return start_bin, end_bin, increment_bin

#Stacked Histogram Function
    def stacked_histogram(self):
        #Import libraries
        import numpy as np
        from matplotlib import pyplot as plt
        #Create combonations of the values for the two options
        data[self.C + "-" + self.D] = data[self.C] + " " + data[self.D]
        combos = np.unique(data[self.C + "-" + self.D])
        #Create variable that we call the function to calculate the bin width
        bin = self.bin_width()
        #Start at value = bin[0], Stop at value = bin[1], Increment by value of bin[2]
        bins = np.array(np.arange(start = bin[0], stop = bin[1], step = bin[2]))
        #Create histogram
        for i in range(len(combos)):
          plt.hist(data[data[self.C+""+self.D].isin(combos[i:(len(combos))])][self.A],
                   bins, label = combos[i:(len(combos))])
        #x-axis label
        plt.xlabel(self.A, fontsize = 16)
        #y-axis lable
        plt.ylabel("Frequency of ", fontsize = 16)
        #Legend of graph
        plt.legend(loc = 'upper left')
        #Title of graph
        plt.title("Histogram of " + self.A + " with unique combinations of " + self.D
              + " and " + self.C, loc = 'center')
        plt.show()
        return

#Create an object from class Graphs that will have three parameters
stacked_histo = Graphs('JOBSAT', None, 'Gender', 'POSITION')
#Call stacked_histogram() function to apply to object
stacked_histo.stacked_histogram()
```

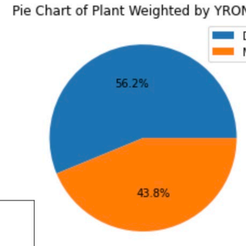


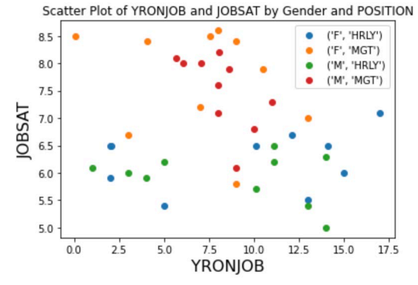




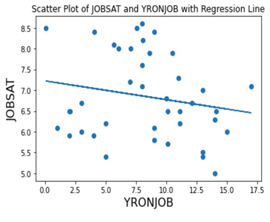


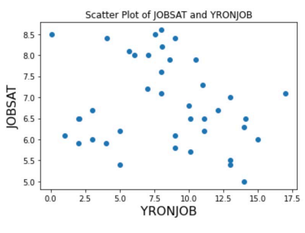

```
The following is listed in descending order according to the legend:
The slope of the line is 0.024201565009048475 and the intercept of the line is 6.036377539316392
The slope of the line is -0.06233537513227938 and the intercept of the line is 8.144451224693151
The slope of the line is -0.0283732339180469 and the intercept of the line is 6.174861008712744
The slope of the line is -0.24706178643384763 and the intercept of the line is 9.526024177300197
```

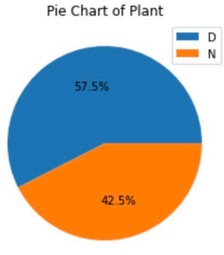





```
The slope of the line is -0.04518720210128208 and the intercept of the line is 7.224601905419628
```







What is a Class?

A class is a user-defined blueprint from which objects are created or instances of the class. Creating a new class creates a new type of object and defines those objects' properties and behaviors. Properties are attributes of an object, and behaviors are methods that modify the object. For example, the class I created, Graphs, has four attributes, two quantitative variables and two qualitative variables, and has ten methods, which two are shown under the sample code section.

Overall, classes provided a simplistic way of keeping attributes and methods together, which helps keep the program organized and allows for reusability. Another functionality of a class is inheritance, or when a defined class inherits all the methods and attributes from another class.

My Code

The class was developed to be completely reproducible and a simplistic experience for the user. All the user has to do is import a cleaned dataset, create an object of the class, pass the desired attributes to the object, and apply a method to the object. The sample code to the left displays two methods and creating an object. The first method calculates the bin width for a histogram, and the second method creates a stacked histogram of a variable with the combination of two other variables. Allowing another method, bin_width(), to calculate the bin width means the user does not have to calculate or define how the bins will increment. The user can simply pass any quantitative variable, and the method will do all the work. Having the bin_width() method be its own method allows other methods to use it as well. The method stacked_histogram() first calculates the possible combinations of two variables. It then calls the method bin_width() to calculate the bins for the graph. Lastly, it creates the histogram by iterating through all possible combinations. Meaning, the user does not have to specify the number of combinations.

Why My Code Matters

Even though data science's impressive side is the more advanced methods like XG Boost and multilayered neural networks, classes are a significant part. Classes allow code to be more readable and reproducible. They also can help users build dashboards to quickly and easily see results and be a part of automated pipelines. The class I developed could be a part of an automated report system for a company with a certain number of standard graphs they want to use to check performance. Using and developing classes, methods, and for loops are all essential parts of a data scientist's job.