WDS Project

Athena Hernandez

August 31, 2022

1 Abstract

For this project, I will be analyzing this dataset titled "2021 Kaggle Machine Learning & Data Science Survey" which is the most comprehensive dataset available on the state of ML and data science. The question I'm exploring is: What factors, like total years coding or salary, have the greatest impact on the gender disparities that lie within data scientists? And how do these factors interact with each other? In this paper, I create a classification model to determine the gender of a user using variables including country, age, salary, and education level. My model worked fairly well with about an average of 75% accuracy. Although this seems low without context, it is important to consider that the majority of my data was heavily based upon my observations on a string-heavy dataset. This meant, I really had to focus on classification and did not have much of an opportunity to explore numerical regression nor classification based upon numerics. Although my model's comparisons to other models struggled with details explained later, comparing my results to many different notebooks across Kaggle, I found similar results confirming my classifier. My classifier can still be improved a lot and I list several ways for how I would go about this later in this paper.

2 Data Exploration

2.1 Setup

[69]:
```
# Import data visualization packages
import pandas as pd
# import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# %matplotlib inline
```

Using pandas's `read_csv()` function, I was able to read in the comma-separated values (CSV) file I downloaded from Kaggle. Because this dataset contained 369 columns, I only picked a few to analyze for this project which I set in the `usecols` parameter. The ones I selected were based on the broad idea that I wanted to analyze gender disparities.

[70]:
```
data = pd.read_csv('survey.csv', usecols = ['Q1', 'Q2', 'Q3', 'Q4', 'Q5', 'Q6',␣
 ↪'Q11', 'Q14_Part_1', 'Q15', 'Q25'], nrows=5000)
```

I also renamed the columns because their titles were not descriptive enough.

[71]:
```
data.columns = ['age', 'gender', 'country', 'education', 'title',␣
↪'years_coding', 'computing_platform', 'libraries', 'years_ml', 'salary']
```

Using Pandas's `head()` function, the first few rows of the dataset that I read in are displayed. As previously stated, the specific columns are now named appropriately and I only have columns that I would like to analyze more in depth.

[72]:
```
data.head()
```

[72]:
```
                          age                                  gender  \
0  What is your age (# years)?  What is your gender? - Selected Choice
1                       50-54                                     Man
2                       50-54                                     Man
3                       22-24                                     Man
4                       45-49                                     Man

                                     country  \
0  In which country do you currently reside?
1                                      India
2                                  Indonesia
3                                   Pakistan
4                                     Mexico

                                           education  \
0  What is the highest level of formal education …
1                                  Bachelor’s degree
2                                    Master’s degree
3                                    Master’s degree
4                                    Doctoral degree

                                               title  \
0  Select the title most similar to your current …
1                                              Other
2                            Program/Project Manager
3                                  Software Engineer
4                                 Research Scientist

                                        years_coding  \
0  For how many years have you been writing code …
1                                         5-10 years
2                                          20+ years
3                                          1-3 years
4                                          20+ years

                                  computing_platform  \
0  What type of computing platform do you use mos…
1                                           A laptop
```

```
2  A cloud computing platform (AWS, Azure, GCP, h…
3                                          A laptop
4  A cloud computing platform (AWS, Azure, GCP, h…

                                         libraries  \
0  What data visualization libraries or tools do …
1                                      Matplotlib
2                                      Matplotlib
3                                      Matplotlib
4                                      Matplotlib

                                          years_ml  \
0  For how many years have you used machine learn…
1                                       5-10 years
2                                     Under 1 year
3              I do not use machine learning methods
4                                       5-10 years

                                            salary
0  What is your current yearly compensation (appr…
1                                    25,000-29,999
2                                    60,000-69,999
3                                           $0-999
4                                    30,000-39,999
```

In order for the graphs to not include the first row of the dataset that contained the questions, I first created a new dataset `data_responses` by using pandas' `iloc` property. I also made sure to reset the index using the `reset_index()` function so pandas knew this dataframe was going to be used completely separately from the old one. I used pandas' `dropna()` function because I wanted to get rid of any rows that contained NaNs as they would disrupt my project.

[73]:
```
data_responses = data.iloc[1:,:].dropna()
data_responses.reset_index(drop=True, inplace=True)
```

Below, you can see that the first row is raw data—not the questions that were at the top earlier—and its index is 0 because I reset it. If I did not reset it, it would be 1 because that is what it was previously.

[74]:
```
data_responses.head()
```

[74]:
```
     age gender    country         education                    title  \
0  50-54    Man      India  Bachelor's degree                    Other
1  50-54    Man  Indonesia    Master's degree  Program/Project Manager
2  22-24    Man   Pakistan    Master's degree        Software Engineer
3  45-49    Man     Mexico   Doctoral degree       Research Scientist
4  45-49    Man      India   Doctoral degree                    Other

  years_coding                              computing_platform  \
```

```
0   5-10 years                                            A laptop
1    20+ years  A cloud computing platform (AWS, Azure, GCP, h…
2    1-3 years                                            A laptop
3    20+ years  A cloud computing platform (AWS, Azure, GCP, h…
4    < 1 years  A cloud computing platform (AWS, Azure, GCP, h…

     libraries                                 years_ml         salary
0   Matplotlib                               5-10 years  25,000-29,999
1   Matplotlib                             Under 1 year  60,000-69,999
2   Matplotlib   I do not use machine learning methods          $0-999
3   Matplotlib                               5-10 years  30,000-39,999
4   Matplotlib                              10-20 years  30,000-39,999
```

Before I can analzye my data using visualization techniques, I need to split my data into training and test data. I did this below by using the Python `random` class to select which rows I will be using in each.

[75]:

```python
# Get total number of data points
num_data_points = len(data_responses)
print(num_data_points)

# Set up an array of all of the row indices
row_indices = np.arange(0, num_data_points)   # List of numbers from 0 to␣
 ↪num_data_points
half_num = num_data_points // 2               # Gets nearest integer after␣
 ↪division by 2

# Randomly selects some row indices for the training data
training_row_indices = np.random.choice(row_indices, half_num, replace = False )

# The rest of the row indices are for the test data:
test_row_indices = np.setdiff1d(row_indices, training_row_indices)

# Pick out the rows of the big dataset based on the chosen row indices
training_data = data_responses.iloc[training_row_indices, :]
test_data = data_responses.iloc[test_row_indices, :]
```

```
2070
```

Below is the the first few lines of the `training_data` dataframe.

[76]:

```python
training_data.head()
```

[76]:

```
        age gender                                           country  \
517   25-29  Woman                          United States of America
2014  30-34    Man  United Kingdom of Great Britain and Northern I…
1462  30-34  Woman                          United States of America
1190  30-34    Man                          United States of America
```

```
1176  60-69    Man                           United States of America

                                         education            title  \
517                                Bachelor’s degree            Other
2014  Some college/university study without earning …    Data Engineer
1462                                 Master’s degree     Data Analyst
1190                               Bachelor’s degree     Data Analyst
1176  Some college/university study without earning …  Product Manager

     years_coding                                 computing_platform  \
517     < 1 years                                           A laptop
2014   5-10 years  A cloud computing platform (AWS, Azure, GCP, h…
1462    < 1 years                                           A laptop
1190    < 1 years                                           A laptop
1176    20+ years  A cloud computing platform (AWS, Azure, GCP, h…

        libraries      years_ml          salary
517    Matplotlib     1-2 years   10,000-14,999
2014   Matplotlib     3-4 years   90,000-99,999
1462   Matplotlib  Under 1 year          $0-999
1190   Matplotlib  Under 1 year   40,000-49,999
1176   Matplotlib     3-4 years 250,000-299,999
```

Below is the the first few lines of the `test_data` dataframe.

[77]:
```
test_data.head()
```

[77]:
```
      age gender    country        education                    title  \
0   50-54    Man      India  Bachelor’s degree                    Other
1   50-54    Man  Indonesia    Master’s degree  Program/Project Manager
2   22-24    Man   Pakistan    Master’s degree        Software Engineer
7   40-44    Man  Australia   Doctoral degree                    Other
12    70+    Man  Singapore  Bachelor’s degree                    Other

   years_coding                               computing_platform  \
0    5-10 years                                         A laptop
1     20+ years  A cloud computing platform (AWS, Azure, GCP, h…
2     1-3 years                                         A laptop
7     1-3 years                      A personal computer / desktop
12    < 1 years                      A personal computer / desktop

      libraries                            years_ml         salary
0    Matplotlib                          5-10 years  25,000-29,999
1    Matplotlib                        Under 1 year  60,000-69,999
2    Matplotlib  I do not use machine learning methods         $0-999
7    Matplotlib  I do not use machine learning methods  70,000-79,999
12   Matplotlib  I do not use machine learning methods  20,000-24,999
```

Now that I've set up my datasets, I am going to begin to understand the problem I want to tackle in this project by creating four data visualizations—each centered around gender. This data visualizations will later on correspond to the same four variables I based my classifier model on because I used these visualizations as an opportunity to work on understanding and brainstorming for my model. Note that for the simplification of my model development as well as due to the amount of data that was submitted by groups other than male and female, I chose to only analyze male and female disparities.

2.2 Data Visualization: Gender Disparities by Country

For my first data visualization, I wanted to analyze the male and female respondents based on their respective country. I achieved this by first finding the top five popular countries in the survey by using the following code.

[78]:
```
# Code from Aram-Alexandre Pooladian
data_country = data.iloc[1:, 2]
data_country_np = data_country.to_numpy()

k=6
country_dict = {}
for i in range(len(data_country_np)):
    gen = data_country_np[i]
    freq = country_dict.get(gen)
    if freq == None:
        country_dict[gen] = 1
    else:
        freq += 1
        country_dict[gen] = freq

k_counter = 0

country_dict_sorted = {k: val for k, val in sorted(country_dict.items(),␣
 ↪key=lambda item: item[1])}
country_dict_sorted_keys_list = list(country_dict_sorted.keys())

countries = [] # Holds most popular countreis
while k_counter < k:
    countries.append(country_dict_sorted_keys_list[-1 - k_counter])
    k_counter += 1
```

The main purpose of code above is to create a list with the top countries that occured in the survey. Due to the way the survey was formatted, the "Other" option ranked third most popular. This is why I extended the list to account for six values, not just top five. This way, I could ignore the third index, "Other", and include the next country on the list, Brazil.

[79]:
```
print(countries)  # Top 6 responses
```

```
['India', 'United States of America', 'Other', 'Japan', 'China', 'Brazil']
```

By using the `remove()` function, Python eliminates the "Other" index in the list `countries`.

[80]:
```
countries.remove("Other")
print(countries)
```

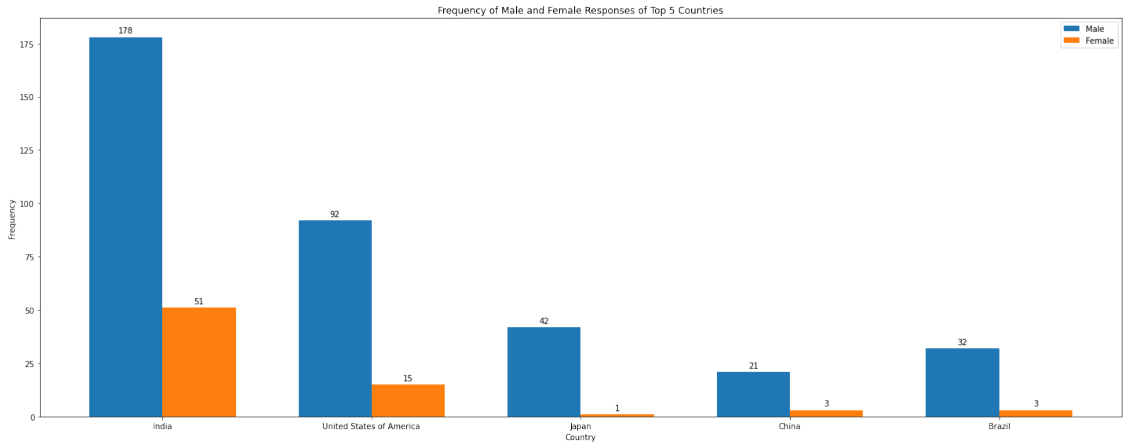
```
['India', 'United States of America', 'Japan', 'China', 'Brazil']
```

Now that I have the top five countries, I want to go through my dataset and find all the responses that match these countries. Then, I want to find how many male and females there are respectively for each country. I do this by creating two empty lists, `male_frequency` and `female_frequency`, and then I go through each country in the list using pandas' `query()` and Python's built-in `len()` function to find how many males and females there actually were. I then put these numbers into `male_total` and `female_total` which I then appended to the lists I created at the top.

[81]:
```
male_frequency = []
female_frequency = []
for country in countries:
    male_total = len(training_data[training_data['country'] == country].
 ↪query('gender == "Man"'))
    female_total = len(training_data[training_data['country'] == country].
 ↪query('gender == "Woman"'))
    male_frequency.append(male_total)
    female_frequency.append(female_total)
```

Now, I have everything I need to plot the bar graph using `Matplotlib`.

[82]:
```
x = np.arange(len(countries))  # Label locations
width = 0.35                   # Width of bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, male_frequency, width, label='Male')
rects2 = ax.bar(x + width/2, female_frequency, width, label='Female')

# Adds text for countries, title, custom x-axis tick countries, and axes
ax.set_ylabel('Frequency')
ax.set_title('Frequency of Male and Female Responses of Top 5 Countries')
ax.set_xlabel('Country')
ax.set_xticks(x, countries)
ax.legend()                    # Male and female color specifications

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

# Resize graph so labels don't overlap
plt.rcParams["figure.figsize"] = (15,5)
plt.show()
```



Not so suprisingly, India, the United States of America, Japan, and China placed in the top five. However, I was not expecting Brazil to have so many data scientists. Because I'm analyzing gender disparities, I took a closer look at the percentage of female users across the countries (respective to order graphed left to right): 23.04%, 22.47%, 5.95%, 11.81%, and 11.11%. Although I can't be for sure that this is the ratio of male to female data scientists for the entire country, it is pretty telling of the fact that women are still struggling to enter into this field of work. The largest ratio of male to female is just below 1:4 which is extremely low. Japan, China, and Brazil have an average ratio of about 1:10 which is even lower than India and the US's ratio. Note that all these numbers were based on the result I got when running my code at the time. Because of the randomness of the training and test data sets, these numbers vary but they are very similar overall.

2.3 Data Visualization: Gender Disparities in Levels of Education

My second data visualization was to take a closer look at the gender disparities by levels of education. I had to use a slightly different approach this time because I needed to find how many unique responses there were first, in order to see how many columns I would be organizing responses into. Using pandas' `unique()` function, I am able to extract all the unique responses and put them into a list I called `education_levels`. This is a key step because it enables me to know how many pairs of columns I will need for my graph. As you can see, after I printed the `education_levels`, there were 7 different choices.

[83]:
```
education_levels = training_data.education.unique() # .education because that␣
↪is the column I want to run unique() on
print(education_levels)
```

```
['Bachelor’s degree'
 'Some college/university study without earning a bachelor’s degree'
 'Master’s degree' 'Doctoral degree' 'Professional doctorate'
 'I prefer not to answer' 'No formal education past high school']
```

Now, similarly to how I gathered the length of the male and female users in my first data visualization, I did the same for this one, except I looked through the `education` column, not the `country` one. Refer to the explanation above.

[84]:
```
male_frequency = []
female_frequency = []
for level in education_levels:
    male_frequency.append(len(training_data.loc[(training_data['gender'] ==␣
 ↪'Man') & (training_data['education'] == level)]))
    female_frequency.append(len(training_data.loc[(training_data['gender'] ==␣
 ↪'Woman') & (training_data['education'] == level)]))
```

Below is the `male_frequency` and `female_frequency`. You can see that these lists are equal in length because they count each male or female, respectively, for each level of education.

[85]:
```
print("\nMale frequency: ", male_frequency, "\nFemale frequency: ",␣
 ↪female_frequency)
```

```
Male frequency:  [277, 39, 376, 133, 9, 19, 8]
Female frequency:  [42, 4, 86, 23, 2, 3, 0]
```

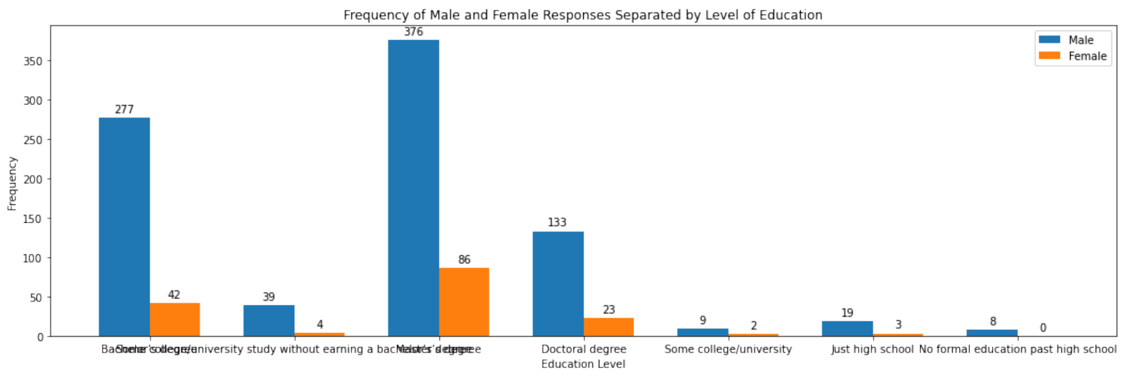
Now, I have all the data I need in order to graph my data. Because some of the responses were really long—specifically index 4 and 5—I shortened its label, so that it would appear cleaner on the graph.

[86]:
```
education_levels[4] = 'Some college/university'
education_levels[5] = 'Just high school'
```

Now, I have everything I need to plot the bar graph using `Matplotlib`.

[87]:
```
x = np.arange(len(education_levels))     # Label locations
width = 0.35                             # Width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, male_frequency, width, label='Male')
rects2 = ax.bar(x + width/2, female_frequency, width, label='Female')

# Adds text for countries, title, custom x-axis tick countries, and axes
ax.set_ylabel('Frequency')
ax.set_title('Frequency of Male and Female Responses Separated by Level of␣
 ↪Education')
ax.set_xlabel('Education Level')
ax.set_xticks(x, education_levels)
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

# Resize graph so labels don't overlap
```

```
plt.rcParams["figure.figsize"] = (20,5)
plt.show()
```



This graph was facinating for many reasons. I was most suprised by the scarcity of doctorates in the data science field. Although I have heard from many people that computer science majors, and similar professions, usually stop before pursuing their doctorate, I did not actually believe that until I saw this graph. I also noticed that the professional doctorate category has the least amount of frequency for both males and females combined. Another trend I noticed was that more females have a master's degree, 396 responses, than a bachelor's degree, 318 responses, while males are more likely to have a bachelor's degree, 1590 responses, than a master's degree, 1527 responses. However, the male discrepancy comparatively between those categories is much less than the female. Because of how little data the latter education levels have, I mostly focused on anaylzing especially bachelor's and master's. Lastly, to analyze the gender disparity numerically, I calculated the percentage female of the total responses for each level of education (respective to order graphed left to right): 18.38%, 18.80%, 18.29%, 21.85%, 18.93%, 16%, and 21.05%. There was a pretty constant trend of about 18%, which is less than a 1:4 ratio.

2.4 Data Visualization: Gender Disparities in Age Groups

My third data visualization was to help me understand the gender disparities broken down by age groups. Similar to how I got the unique values for the education levels, I found all the age groups that the datset provided and put them into a list called `ages`.

[88]:
```
ages = training_data.age.unique()
print(ages)
```

```
['25-29' '30-34' '60-69' '35-39' '50-54' '45-49' '22-24' '40-44' '55-59'
 '18-21' '70+']
```

Because these values are strings and not numbers, I ordered them manually because it would take too much time to parse and organize them by converting them to integers. After I did my best to rearrange, I reset this list in the same list variable, `ages`.

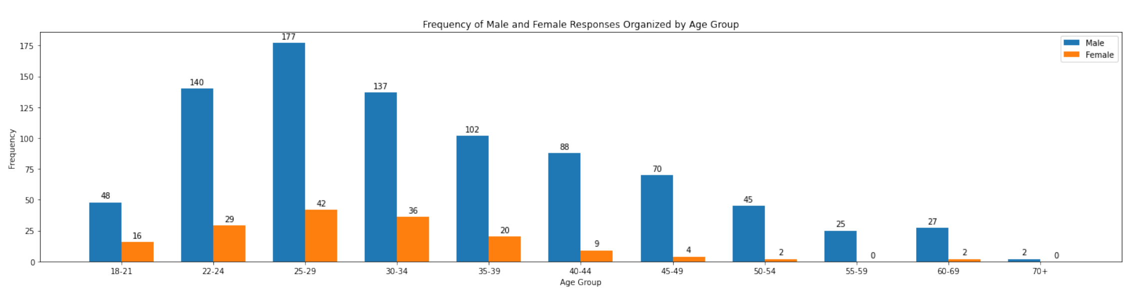
[89]:
```
ages = ['18-21', '22-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54',␣
↪'55-59', '60-69', '70+']
```

```python
print(ages)
```

```
['18-21', '22-24', '25-29', '30-34', '35-39', '40-44', '45-49', '50-54',
'55-59', '60-69', '70+']
```

Following the same steps I did for the previous data visualization, I created and found the values for the `male_frequency` list and the `female_frequency` list. After that, I had all the information I needed in order to graph.

[90]:
```python
male_frequency = []
female_frequency = []
for age in ages:
    male_frequency.append(len(training_data.loc[(training_data['gender'] ==␣
↪'Man') & (training_data['age'] == age)]))
    female_frequency.append(len(training_data.loc[(training_data['gender'] ==␣
↪'Woman') & (training_data['age'] == age)]))

x = np.arange(len(ages))  # Label locations
width = 0.35              # Width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, male_frequency, width, label='Male')
rects2 = ax.bar(x + width/2, female_frequency, width, label='Female')

# Adds text for countries, title, custom x-axis tick countries, and axes
ax.set_ylabel('Frequency')
ax.set_title('Frequency of Male and Female Responses Organized by Age Group')
ax.set_xlabel('Age Group')
ax.set_xticks(x, ages)
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

# Resize graph so labels don't overlap
plt.rcParams["figure.figsize"] = (25,5)
plt.show()
```



While men still take the lead by a large amount in every age group shown in this bar graph, it is important to focus on the younger age groups because that is where the data is the most concentrated for both male and female. Younger and future generations' trends can also help us understand what we are doing well and what we can improve for disparities within gender. Noticiably, for both male and female, age groups 18-21, 22-24, and 25-29 are the peak of the responses for this survey; for males, the frequency of responses is always above 700 and for females, the frequency of responses is always above 180 in these age groups. Approximately 28.12% responses of age group 18-21 were female. This percentage is much more than the 50-54 age group which had only 12.44% female responses. Considering the other age groups follows this trend, I took away from this graph that the gender gap is closing, but at a slow and steady rate. Similar to how I analyzed the other graphs, I do the same by calculating what percentage of each age group is female (respective to order graphed left to right): 21.95%, 19.85%, 20.69%, 19.29%, 19.79%, 16.42%, 11.06%, 11.52%, 5.61%, 5.13%, and 7.69%.

2.5 Data Visualization: Gender Disparities in Salaries

My fourth data visualization was to help me understand the gender disparities broken down by salaries. Similar to how I got the unique values for the education levels and age groups, I found all the salary options that the datset provided and put them into a list called `salaries`.

[91]:
```
salaries = training_data.salary.unique() # education_data[education].unique()
print(salaries)
```

```
['10,000-14,999' '90,000-99,999' '$0-999' '40,000-49,999'
 '250,000-299,999' '300,000-499,999' '15,000-19,999' '125,000-149,999'
 '4,000-4,999' '25,000-29,999' '150,000-199,999' '7,500-9,999'
 '60,000-69,999' '100,000-124,999' '5,000-7,499' '30,000-39,999'
 '1,000-1,999' '20,000-24,999' '50,000-59,999' '3,000-3,999'
 '80,000-89,999' '200,000-249,999' '70,000-79,999' '2,000-2,999'
 '>$1,000,000' '$500,000-999,999']
```

Due to the fact that these salaries weren't in numerical order, I rearranged the order, so that the next step would fill up the `male_frequency` and `female_frequency` list in an order that makes sense when looking at the graph. I did this manually because it was easier than going through the list and changing their type from strings to floats which could only be done after I parsed through each number to find where the first number ends. Although some have dollar signs and others don't, I make sure to leave it that way so that when I go through the dataframe, these will be recognized as is. It is only after I go through the dataframe that I can rename these labels. I also ignored `nan` because that just lets me know that the user did not fill out this question.

[92]:
```
salaries = ['$0-999', '1,000-1,999', '2,000-2,999', '3,000-3,999',␣
 ↪'4,000-4,999', '5,000-7,499', '7,500-9,999',
            '10,000-14,999', '15,000-19,999', '20,000-24,999', '25,000-29,999',␣
 ↪'30,000-39,999', '40,000-49,999',
            '50,000-59,999', '60,000-69,999', '70,000-79,999', '80,000-89,999',␣
 ↪'90,000-99,999', '100,000-124,999',
```

```
            '125,000-149,999', '150,000-199,999', '200,000-249,999',␣
↪'250,000-299,999', '300,000-499,999',
            '$500,000-999,999', '>$1,000,000']
print(salaries)
```

```
['$0-999', '1,000-1,999', '2,000-2,999', '3,000-3,999', '4,000-4,999',
'5,000-7,499', '7,500-9,999', '10,000-14,999', '15,000-19,999', '20,000-24,999',
'25,000-29,999', '30,000-39,999', '40,000-49,999', '50,000-59,999',
'60,000-69,999', '70,000-79,999', '80,000-89,999', '90,000-99,999',
'100,000-124,999', '125,000-149,999', '150,000-199,999', '200,000-249,999',
'250,000-299,999', '300,000-499,999', '$500,000-999,999', '>$1,000,000']
```

Using the same method as before, I get all the frequencies for male and female and put them into `male_frequency` and `female_frequency` respectively.

[93]:
```
male_frequency = []
female_frequency = []
for salary in salaries:
    male_frequency.append(len(training_data.loc[(training_data['gender'] ==␣
↪'Man') & (training_data['salary'] == salary)]))
    female_frequency.append(len(training_data.loc[(training_data['gender'] ==␣
↪'Woman') & (training_data['salary'] == salary)]))
print("\nMale frequency: ", male_frequency, "\nFemale frequency: ",␣
↪female_frequency)
```

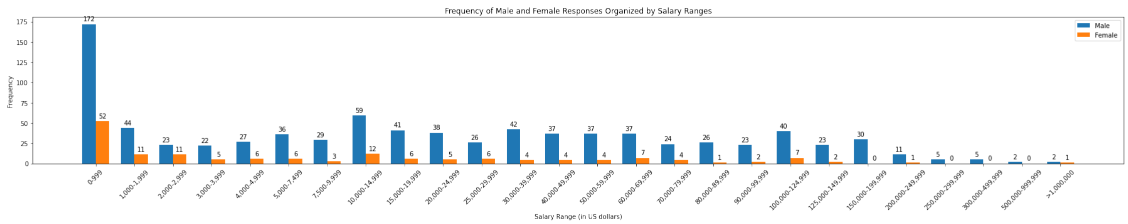
```
Male frequency:  [172, 44, 23, 22, 27, 36, 29, 59, 41, 38, 26, 42, 37, 37, 37,
24, 26, 23, 40, 23, 30, 11, 5, 5, 2, 2]
Female frequency:  [52, 11, 11, 5, 6, 6, 3, 12, 6, 5, 6, 4, 4, 4, 7, 4, 1, 2, 7,
2, 0, 1, 0, 0, 0, 1]
```

Below I rename the labels with dollar signs so that they all follow the same no-units standard.

[94]:
```
salaries[0] = '0-999'
salaries[24] = '500,000-999,999'
salaries[25] = '>1,000,000'
```

[95]:
```
x = np.arange(len(salaries))  # Label locations
width = 0.35                  # Width of the bars

fig, ax = plt.subplots()
rects1 = ax.bar(x - width/2, male_frequency, width, label='Male')
rects2 = ax.bar(x + width/2, female_frequency, width, label='Female')

# Adds text for countries, title, custom x-axis tick countries, and axes
ax.set_ylabel('Frequency')
ax.set_title('Frequency of Male and Female Responses Organized by Salary␣
↪Ranges')
```

```
ax.set_xlabel('Salary Range (in US dollars)')
ax.set_xticks(x, salaries)
plt.xticks(rotation = 45)     # Rotate x-axis labels b/c didn't fit horizontally
ax.legend()

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

fig.tight_layout()

# Resize graph so labels don't overlap
plt.rcParams["figure.figsize"] = (20,8)
plt.show()
```



This graph was very intriguing to me for multiple reasons. The first being that the majority of the responses were by users who were making less than $1000 per year. Next, it was interesting that the male frequency varies quite a lot in comparison to the female responses that seemed pretty constant after reaching the '2,000-2,999' range. There's also an interesting spike in the '100,000-124,999' range that was unexpected. Although this feature of the dataframe is not very telling of whether a user is male or female, it will definitely help with the building of my classifier.

3 Model Development

My goal for developing this model was to determine if a user was male or female based on their country, education level, salary, and age group. For my model, I wanted to implement the knowledge I had gained from analyzing the different data visualizations I learned about above. Because my data was based more upon categories, I thought it would be best to implement a classification model. After analyzing the graphs, I was able to put together a classifier model based on large trends I noticed. One big struggle I encountered was the fact that a majority of the responses were male which made it difficult to classify; however, finding the trends that I did and breaking that down into the features that were provided by the dataframe, I as able to put together a model to determine if a user was male or female. Something to note was that it was difficult to gather an understanding of exactly how to build my classifier simply based on the graphs above in the Data Visualiation section. Instead, I also played around with the training dataframe itself to gather a better understanding.

The first thing I needed to do was to create a smaller dataframe with only the features I wanted to include now that I was definitive of them: country, education level, salary, and age group.

[96]:
```
model_data = training_data.iloc[:, [0, 1, 2, 3, 9]]
model_data.head()
```

[96]:
```
        age gender                                              country  \
517   25-29  Woman                             United States of America
2014  30-34    Man  United Kingdom of Great Britain and Northern I…
1462  30-34  Woman                             United States of America
1190  30-34    Man                             United States of America
1176  60-69    Man                             United States of America

                                               education           salary
517                                    Bachelor's degree    10,000-14,999
2014  Some college/university study without earning …    90,000-99,999
1462                                     Master's degree           $0-999
1190                                   Bachelor's degree    40,000-49,999
1176  Some college/university study without earning …  250,000-299,999
```

As I previously mentioned, in order to gain a better understanding of the material because the graphs sometimes didn't do enough for me to understand the complex relationships between the four variables I chose, I needed to play around with the training dataframe itself. Below is an example of me taking a look at the ratios of males versus females in China organized by education levels.

[97]:
```
print("Bachelor's degree")
print("M:", len(training_data.loc[(training_data['gender'] == 'Man') &␣
 ↪(training_data['education'] == 'Bachelor's degree') &␣
 ↪(training_data['country'] == 'China')]) / len(training_data.
 ↪loc[(training_data['country'] == 'China')]))
print("F:", len(training_data.loc[(training_data['gender'] == 'Woman') &␣
 ↪(training_data['education'] == 'Bachelor's degree') &␣
 ↪(training_data['country'] == 'China')]) / len(training_data.
 ↪loc[(training_data['country'] == 'China')]))

print("\nMaster's degree")
print("M:", len(training_data.loc[(training_data['gender'] == 'Man') &␣
 ↪(training_data['education'] == 'Master's degree') &␣
 ↪(training_data['country'] == 'China')]) / len(training_data.
 ↪loc[(training_data['country'] == 'China')]))
print("F:", len(training_data.loc[(training_data['gender'] == 'Woman') &␣
 ↪(training_data['education'] == 'Master's degree') &␣
 ↪(training_data['country'] == 'China')]) / len(training_data.
 ↪loc[(training_data['country'] == 'China')]))

print("\nDoctoral degree")
```

```
print("M:", len(training_data.loc[(training_data['gender'] == 'Man') &␣
↪(training_data['education'] == 'Doctoral degree') &␣
↪(training_data['country'] == 'China')]) / len(training_data.
↪loc[(training_data['country'] == 'China')]))
print("F:", len(training_data.loc[(training_data['gender'] == 'Woman') &␣
↪(training_data['education'] == 'Doctoral degree') &␣
↪(training_data['country'] == 'China')]) / len(training_data.
↪loc[(training_data['country'] == 'China')]))

print("\nSome college/university study without earning a bachelor’s degree")
print("M:", len(training_data.loc[(training_data['gender'] == 'Man') &␣
↪(training_data['education'] == 'Some college/university study without␣
↪earning a bachelor’s degree') & (training_data['country'] == 'China')]) /␣
↪len(training_data.loc[(training_data['country'] == 'China')]))
print("F:", len(training_data.loc[(training_data['gender'] == 'Woman') &␣
↪(training_data['education'] == 'Some college/university study without␣
↪earning a bachelor’s degree') & (training_data['country'] == 'China')]) /␣
↪len(training_data.loc[(training_data['country'] == 'China')]))
```

```
Bachelor’s degree
M: 0.12
F: 0.0

Master’s degree
M: 0.6
F: 0.08

Doctoral degree
M: 0.04
F: 0.04

Some college/university study without earning a bachelor’s degree
M: 0.08
F: 0.0
```

Below is the actual code to my classifier model. The parameters of this function are `xcountry`, `xeducation`, `xage`, and `xsalary`—the features I am basing the model upon. `gender_classifier()` will return either 1 or 0 if it classifies the user as male or female respectively. As for how I branched my classifier, it really took a lot of attention to detail similar to the what I did in the previous step. However, I started out by looking at the graphs I made in the Data Visualization section.

[98]:
```
# X train has age, country, degrees, salary (all strings)
def gender_classifier(xcountry, xeducation, xage, xsalary):
    if xcountry == 'India':
        if xeducation == 'Master’s degree':
            return 0        # Female
        elif xsalary in ['$0-999', '1,000-1,999']:
                return 0
```

```
        else:
            if age in ['18-21', '22-24', '35-39']:
                return 0
            else:
                return 1    # Male
    elif xcountry == 'United States of America':
        if xeducation == 'Master’s degree':
            return 1
        else:
            if xsalary in ['$0-999', '1,000-1,999', '2,000-2,999']:
                return 0
            elif age in ['18-21', '22-24', '35-39']:
                return 0
            return 1
    elif xcountry == 'China':
        if xeducation in ['Bachelor’s degree', 'Some college/university study␣
↪without earning a bachelor’s degree']:        # No female Bachelor degrees␣
↪in China
            return 1
        else:
            if xsalary in ['$0-999', '1,000-1,999']:
                return 0
            else:
                if age in ['18-21', '22-24', '35-39']:
                    return 0
                else:
                    return 1
    else:
        return 1
```

Below are a few examples of the model in action. Recall that if it returns a 1, then the function is classifying it as a male and if it returns a 0, then the function is classifying it as a female.

[99]:
```
gender_classifier('United States of America', 'Master’s degree', '18-21',␣
↪'20,000-29,999')
```

[99]: 1

[100]:
```
gender_classifier('China', 'Master’s degree', '18-21', '1,000-1,999')
```

[100]: 0

4 Assessment of Model

The following code creates a new list `data_model_outcomes` and goes through the dataset to assign a value of 0 or 1 for female and male responses respectively.

[101]:
```
model_data_outcomes = []
for i in model_data.iloc[:,1]:
    if i == 'Man':
        model_data_outcomes.append(1)
    elif i == 'Woman': # Excluding other values (ie nonbinary) b/c limited␣
↪amount of data
        model_data_outcomes.append(0)
```

I summed the list below because now you can see that there are that many male responses in the training dataset. The 1s that are being added are the 1s that were assigned due to the fact that the response was a male user.

[102]:
```
sum(model_data_outcomes)
```

[102]:
```
861
```

In order to test my model, I went ahead and created a testing and training dataset—one for input, one for output. In the output is the `X_test`, which is a random selection of data used for testing purposes.

[109]:
```
from sklearn.model_selection import train_test_split
X = test_data
Y = test_data['gender']

X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.3,␣
↪random_state = 1)

X_test.head()
```

[109]:
```
        age gender country                                          education  \
1739  25-29    Man   Japan                                    Master’s degree
1164  22-24    Man   Spain  Some college/university study without earning …
551   18-21    Man   India                                  Bachelor’s degree
2008  40-44    Man   India                                    Master’s degree
209   55-59    Man  Sweden                                    Doctoral degree

                        title years_coding             computing_platform  \
1739       Research Scientist    3-5 years                       A laptop
1164           Data Scientist    3-5 years                       A laptop
551            Data Scientist    1-3 years                       A laptop
2008         Business Analyst   5-10 years                       A laptop
209   Program/Project Manager    20+ years  A personal computer / desktop

       libraries      years_ml          salary
1739  Matplotlib     2-3 years   40,000-49,999
1164  Matplotlib     3-4 years   10,000-14,999
551   Matplotlib     1-2 years          $0-999
2008  Matplotlib  Under 1 year     4,000-4,999
```

```
209    Matplotlib      1-2 years  70,000-79,999
```

These functions compare the actual value and predicted value in various ways. The first function, `is_wrong()` calulates whether they `y_a` and `y_p` match. The rest of the functions, `true_pos()`, `true_neg()`, `false_pos()`, and `false_neg()` are described in comments down below.

[110]:
```
def is_wrong(y_a,y_p):
    if y_a == y_p:     # If actual equals predicted
        return 0
    else:
        return 1

def true_pos(y_a,y_p):
    if (y_a == 1):     # If actual label is positive
        if (y_p == 1): # And predicted label is positive
            return 1   # True positive is TRUE
        else:
            return 0
    else:
        return 0

def true_neg(y_a,y_p):
    if (y_a == 0):     # If actual label is negative
        if (y_p == 0): # And predicted label is negative
            return 1   # True negative is TRUE
        else:
            return 0
    else:
        return 0
def false_pos(y_a,y_p):
    if (y_a == 0):     # If actual label is negative
        if (y_p == 1): # And predicted label is positive
            return 1   # False positive is TRUE
        else:
            return 0
    else:
        return 0
def false_neg(y_a,y_p):
    if (y_a == 1):      # If actual label is positive
        if (y_p == 0):  # And predicted label is negative
            return 1    # False negative is TRUE
        else:
            return 0
    else:
        return 0
```

Here is where I actually compare my predictions to the actual value. I do this by creating a new dataframe called `predictions_df` which I fill up with values specified, what the functions above return, and display that in the output when I print the dataframe's head.

[111]:
```
# PREDICT THE CLASS OF EACH ROW OF THE TEST DATASET, USING A FOR LOOP

rows_test = len(X_test)

# first make a blank data frame to record our predictions
predictions_df = pd.DataFrame( np.empty( ( rows_test  ,  7 ) ) )
predictions_df.rename( columns = {0:'actual',
                                  1:'predicted',
                                  2:'error',
                                  3:'tp',
                                  4:'tn',
                                  5:'fp',
                                  6:'fn'} , inplace = True )

rows = np.arange(0, rows_test)

for row in rows:
    # Make predictions for each row of test dataset
    y_p = gender_classifier(X_test.iloc[row, 2], X_test.iloc[row, 3], X_test.
 ↪iloc[row, 0], X_test.iloc[row, 9])

    if y_test.iloc[row] == 'Man':
        y_a = 1
    elif y_test.iloc[row] == 'Woman': # Used elif b/c nonbinary + others not␣
 ↪included in classification
        y_a = 0

    # Place in dataframe
    predictions_df.iloc[row, 0] = y_a
    predictions_df.iloc[row, 1] = y_p

    # Fill out error column
    predictions_df.iloc[row, 2] = is_wrong(y_a, y_p)

    # Fill out tp, tn, fp, fn columns
    predictions_df.iloc[row, 3] = true_pos(y_a, y_p)
    predictions_df.iloc[row, 4] = false_pos(y_a, y_p)
    predictions_df.iloc[row, 5] = true_neg(y_a, y_p)
    predictions_df.iloc[row, 6] = false_neg(y_a, y_p)

predictions_df.head(15)
```

[111]:

	actual	predicted	error	tp	tn	fp	fn
0	1.0	1.0	0.0	1.0	0.0	0.0	0.0
1	1.0	1.0	0.0	1.0	0.0	0.0	0.0

2	1.0	0.0	1.0	0.0	0.0	0.0	1.0
3	1.0	0.0	1.0	0.0	0.0	0.0	1.0
4	1.0	1.0	0.0	1.0	0.0	0.0	0.0
5	1.0	1.0	0.0	1.0	0.0	0.0	0.0
6	1.0	1.0	0.0	1.0	0.0	0.0	0.0
7	1.0	1.0	0.0	1.0	0.0	0.0	0.0
8	1.0	0.0	1.0	0.0	0.0	0.0	1.0
9	1.0	1.0	0.0	1.0	0.0	0.0	0.0
10	1.0	1.0	0.0	1.0	0.0	0.0	0.0
11	1.0	1.0	0.0	1.0	0.0	0.0	0.0
12	1.0	1.0	0.0	1.0	0.0	0.0	0.0
13	1.0	1.0	0.0	1.0	0.0	0.0	0.0
14	1.0	1.0	0.0	1.0	0.0	0.0	0.0

Now that I have my `predictions_df` all filled out and complete, I can assess my model with two performance metrics. The first being accuracy and the second being the true negative rate. Because this classification model doesn't use machine learning that falls into percentages, that makes it difficult to find a performance metric that aligns with the "top five" idea, so I just analyzed the true positive.

[112]:
```
num_errors = np.sum(predictions_df['error'])
error_rate = np.mean(predictions_df['error'])
accuracy = 1 - error_rate

num_tp = np.sum(predictions_df['tp'])
num_fp = np.sum(predictions_df['fp'])
num_tn = np.sum(predictions_df['tn'])
num_fn = np.sum(predictions_df['fn'])
num_p = num_fn + num_tp
num_n = num_fp + num_tn

tn_rate = num_tn / (num_fp + num_tn)     # Negative rate

print('Accuracy:', accuracy)
print('True negative rate:', tn_rate)
```

```
Accuracy: 0.7652733118971061
True negative rate: 0.75
```

Although these values are not as close to 100% as desirable, I think my classifier did decently well. Considering the fact that my dataset was very open ended and had various nooks and cranies to analyze, 79.01% accuracy with a 75.56% true negative rate is pretty good. In the future, I could raise my accuracy by looking more in depth at each layer of my classifier model and be more conscious of exactly what I classify and where. I found the accuracy by subtracting the error rate from 1 and later the true negative rate using `tn / (fp + tn)`.

5 Comparison to Expert Models

Although I wanted to use a KNN model, my dataset proved difficult for me to apply it to. The `fit()` function doesn't work with my data because it wants floats, not strings. I could go about this by assigning each unique value a corresponding number, but that takes too much time and variability to account for, especially with the random test dataset. If I were to choose a k it would be about 10 because of the mere size of my dataset. Maybe that number would shift a bit, but the main idea is that I would have to find a number that can still be specifically sensitive enohugh to where a delicate classification can still be made within the very large dataset presented. I would reanalzye my accuracy and true negative rates to see how the KNN classifier produced a different response than my own classifier.

[116]:
```
"""
from sklearn.neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors = 10)
model.fit(X_train, y_train)
y_p = model.predict(X_test)

accuracy_knn = model.score(X_test, y_test)
print("KNN accuracy:", accuracy_knn)

model.predict(X_test)

for row in rows:
    y_actual = y_tests[row]
    predictions_knn.iloc[row, 0] = y_pred[row]
    predictions_knn.iloc[row, 1] = y_actual
    predictions_knn.iloc[row, 2] = is_wrong(y_actual, y_pred[row])
    predictions_knn.iloc[row, 3] = true_pos(y_actual, y_pred[row])
    predictions_knn.iloc[row, 4] = true_neg(y_actual, y_pred[row])
    predictions_knn.iloc[row, 5] = false_pos(y_actual, y_pred[row])
    predictions_knn.iloc[row, 6] = false_neg(y_actual, y_pred[row])

num_errors = np.sum( predictions_knn['error'])
error_rate = np.mean( predictions_knn['error'])
accuracy_knn = model.score(X_tests, y_tests)

num_tp_knn = np.sum( predictions_knn['tp'])
num_fp_knn = np.sum( predictions_knn['fp'])
num_tn_knn = np.sum( predictions_knn['tn'])
num_fn_knn = np.sum( predictions_knn['fn'])
num_p_knn = num_fp_knn + num_tp_knn
num_n_knn = num_fn_knn + num_tn_knn

precision_knn = num_tp_knn/ num_p_knn
tp_rate_knn = num_tp_knn / (num_tp_knn + num_fn_knn)
tn_rate_knn = num_tn_knn / (num_fn_knn + num_tn_knn)
```

```
print('Accuracy with K= 10:', accuracy_knn)
"""
```

[116]:
```
'\nfrom sklearn.neighbors import KNeighborsClassifier\n\nmodel =
KNeighborsClassifier(n_neighbors = 10)\nmodel.fit(X_train, y_train)\ny_p =
model.predict(X_test)\n\naccuracy_knn = model.score(X_test, y_test)\nprint("KNN
accuracy:", accuracy_knn)\n\nmodel.predict(X_test)\n\nfor row in rows:\n
y_actual = y_tests[row]\n    predictions_knn.iloc[ row, 0 ] = y_pred[row]\n
predictions_knn.iloc[ row , 1 ] = y_actual\n    predictions_knn.iloc[row,2] =
is_wrong(y_actual,y_pred[row])\n    predictions_knn.iloc[row,3] =
true_pos(y_actual,y_pred[row])\n    predictions_knn.iloc[row,4] =
true_neg(y_actual,y_pred[row])\n    predictions_knn.iloc[row,5] =
false_pos(y_actual,y_pred[row])\n    predictions_knn.iloc[row,6] =
false_neg(y_actual,y_pred[row])\n\nnum_errors = np.sum(
predictions_knn[\'error\'])\nerror_rate = np.mean(
predictions_knn[\'error\'])\naccuracy_knn = model.score(X_tests,
y_tests)\n\nnum_tp_knn = np.sum( predictions_knn[\'tp\'])\nnum_fp_knn = np.sum(
predictions_knn[\'fp\'])\nnum_tn_knn = np.sum(
predictions_knn[\'tn\'])\nnum_fn_knn = np.sum(
predictions_knn[\'fn\'])\nnum_p_knn = num_fp_knn + num_tp_knn\nnum_n_knn =
num_fn_knn + num_tn_knn\n\nprecision_knn = num_tp_knn/ num_p_knn\ntp_rate_knn =
num_tp_knn / (num_tp_knn + num_fn_knn)\ntn_rate_knn = num_tn_knn / (num_fn_knn +
num_tn_knn)\nprint(\'Accuracy with K= 10:\', accuracy_knn)\n'
```

6 Human Context Discussion

Gender disparities in any STEM related field are a large problem that each younger generation is tackling from a young age. Whether this be due to the mere amount of access to STEM classes during elementary school, or the amount of irreversible damage that the past has inflicted upon our current way of living, using data to analyze this is key. By finding common themes and trends—especially for the more recent age groups—we can guide future generations in a better direction than our current one.

There are many sides to this issue based on what people find most important to take action on. Affirmative action based on sex? A plethora of STEM classes mandatory to be taught in every public school? Equal opportunities? How do we define equal? This never-ending argument is continued every day, but by analyzing the data and findind trends about where women in STEM lies in the future is a powerful thing. Although this dataset only covers specifically "data scientists," a future project could be to take a closer look at the differences within the STEM areas of study and see which is being more affected than others by the gender disparities.

Many other curious learners liked me have used this dataset to explore many questions which goes to show that there is so much to unpack from just a signle data set. These predictive models can additionally help users of the dataset to identify where they can improve in. It also gives them a more wholistic view of their work's meaning.

Thank you to the Winston Foundation, GSTEM faculty and peers at the New York University

Courant Institute of Mathematical Sciences, and Aram-Alexandre Pooladian for providing me with this opportunity this summer. I learned a lot and am bound to apply that newfound knowledge in university and in my future career.

References: Kaggle. (2022, January). 2021 Kaggle Machine Learning & Data Science Survey. Kaggle. Retrieved August 31, 2022, from https://www.kaggle.com/competitions/kaggle-survey-2021/data?select=kaggle_survey_2021_responses.csv