Problem and Significance

The problem this paper intends to address is the amount of misinformation being spread through the internet about what is “required” to be a successful software developer. Different people will tell you different things about whether going to college is required, what industry you have to be in, or whether you need to become a manager to succeed. With all of these opinions out there, I want to use data to prove whether any of these “rules” are actually important.

This problem is important because this misinformation can impact people’s lives. Young adults who gain an interest in software development might research online looking for guidance about how to turn their interests into a career. If their research leaves them with a checklist of random things that they need to do (do 1000 leetcode problems, learn 18 languages, use these 7 javascript frameworks, and only apply to FAANG), then they could get discouraged from pursuing software engineering as a career. In reality, software development is a very diverse field, with a diverse range of jobs available. Having objective data showing that these checklists aren’t necessary can allow young programmers to continue to pursue their interest, without being discouraged by irrelevant opinions.

The data I am using to solve this problem comes from the 2024 Stack Overflow Developer Survey, which is, to the best of my knowledge, the largest publicly available survey of software developers. It contains 65,000 responses, with roughly 113 questions. There are questions on everything from technology preferences to workplace structure, background, and job satisfaction.

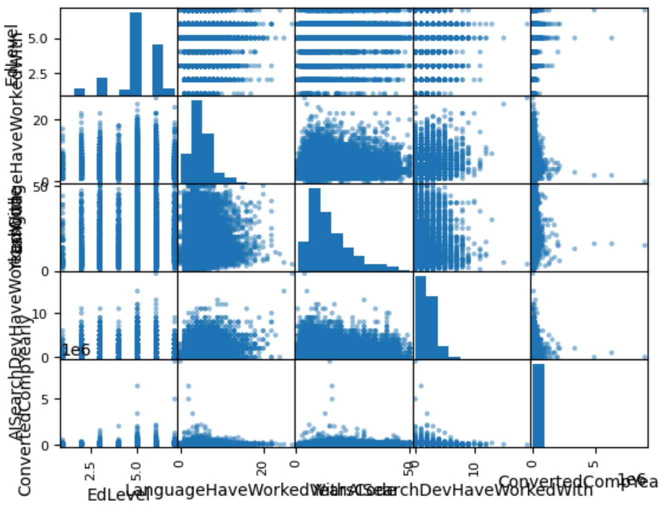
Exploratory Data Analysis

I’ve done a fair amount of data exploration with this dataset during previous assignments, and while investigating different possible questions to ask for this final project. This scatter matrix was one of the first things I looked at after cleaning my data to get a quick view to check that there wasn’t too much collinearity or other major issues. As you can see, there are some significant outliers in the total compensation column, which has probably impacted previous assignments, but this project doesn’t directly do any training with salary and thus shouldn’t be impacted by these outliers.

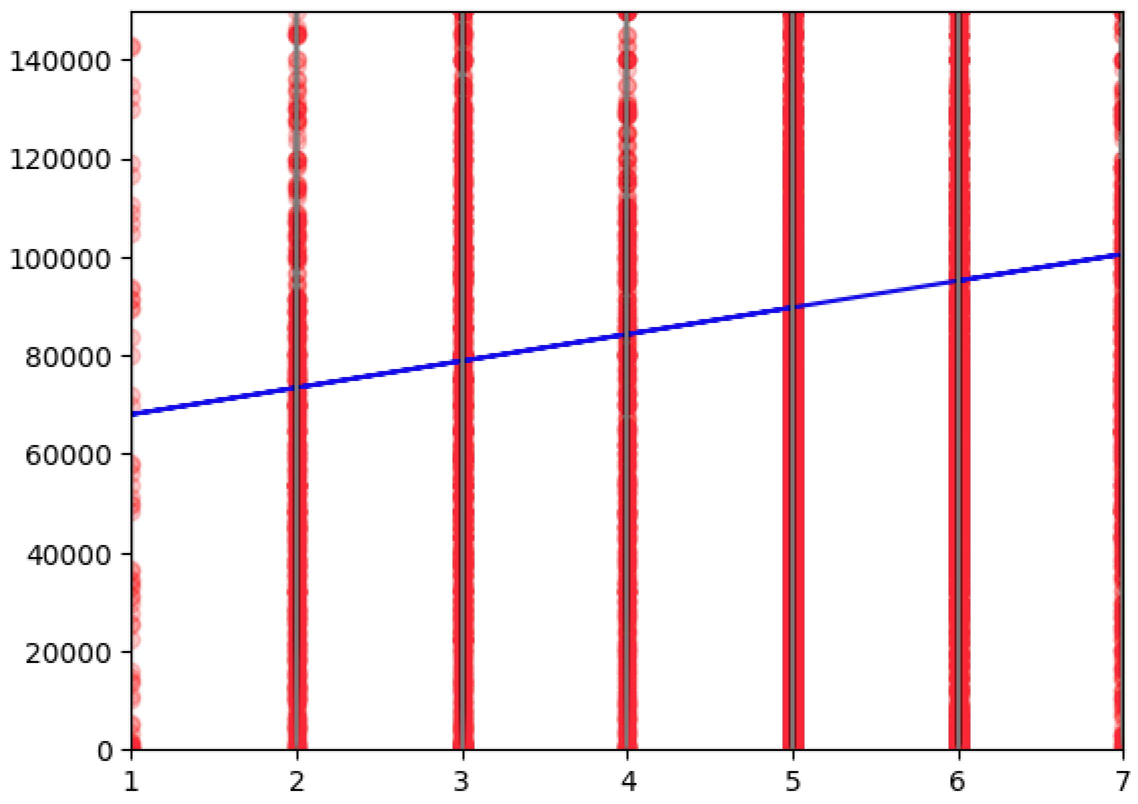
[75]:

	EdLevel	LanguageHaveWorkedWith	YearsCode	AISearchDevHaveWorkedWith	ConvertedCompYearly
count	16112.000000	16112.000000	16112.000000	16112.000000	1.611200e+04
mean	4.996711	5.092912	15.069513	1.659384	8.968860e+04
std	1.159301	2.814765	9.593074	1.487012	1.349697e+05
min	1.000000	0.000000	1.000000	0.000000	1.000000e+00
25%	5.000000	3.000000	8.000000	1.000000	3.866600e+04
50%	5.000000	5.000000	12.000000	1.000000	7.000000e+04
75%	6.000000	7.000000	20.000000	2.000000	1.123558e+05
max	7.000000	27.000000	50.000000	19.000000	9.000000e+06

```
[77]: _ = pd.plotting.scatter_matrix(data)
```



Additionally from previous assignments, I have a few regression models that have shown the relative importance of the above factors on total salary. From most to least important, we have education level, years coding, and number of languages worked with, with the number of AI tools used having a negative impact on compensation. However, when looking at how individual variables impacted total compensation, there was a very low R-squared statistic, showing that education level alone explained very little variability.



The large quantity of data that I have available here makes it difficult to tell visually whether
graph do look decidedly non-linear, but this was a data encoding that had already assumed v
there were outliers with salaries north of $8 million or how many would be over $1 million.

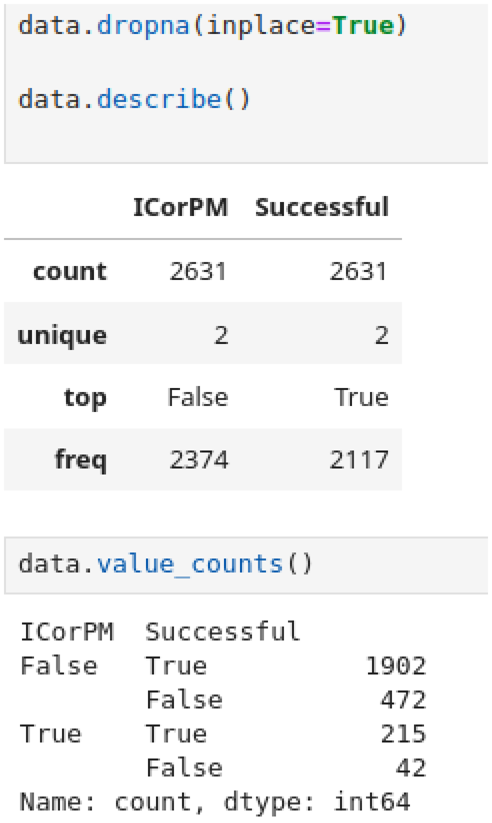
```
def model_eval(pred, y):
    rss = np.sum((pred - y)**2)
    rse = (rss/(len(y) - 2))**(1/2)
    tss = np.sum((y - np.mean(y))**2)
    r2 = 1 - rss/tss
    return rss, rse, tss, r2

_, rse, _, r2 = model_eval(comp_pred, comp)
print("RSE:", rse)
print("R^2:", r2)
```

```
RSE: 134827.29218921837
R^2: 0.0021703059340044684
```

For this project, I didn't have any issues with missing, normalized, or imbalanced data. The survey was large enough that there were plenty of datapoints so I could just ignore any responses that had NA in a field that I cared about. I had to do a fair amount of cleaning of the data before it was usable for some purposes, because most of the fields were in text, but once that was completed, the data was fairly easy to use.

Model Selection, Application, and Evaluation

I may have asked multiple questions in my introduction, but they are all fairly simple to answer. To establish that there is no connection between a predictor and a response, I only need to show that the response variable doesn't change regardless of the predictor values. Two of my questions can be distilled down to a single binary predictor and a single binary response. In this simple case, I can just calculate the proportion of the True response (as opposed to the False response) in the dataset given one predictor value, and then again with the other predictor value. If the response doesn't change, then the values are independent of each other.

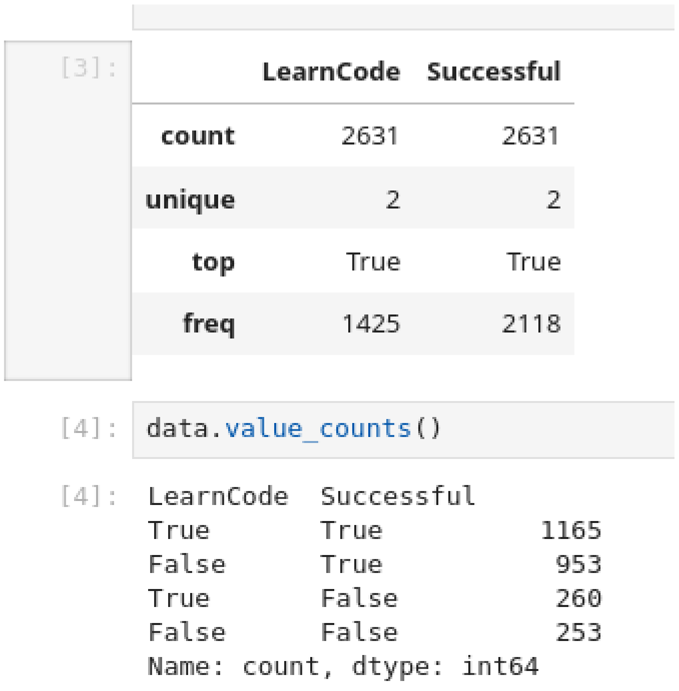


```
data.dropna(inplace=True)

data.describe()
```

	ICorPM	Successful
count	2631	2631
unique	2	2
top	False	True
freq	2374	2117

```
data.value_counts()
```

```
ICorPM  Successful
False   True          1902
        False          472
True    True           215
        False           42
Name: count, dtype: int64
```



```
[3]:
```

	LearnCode	Successful
count	2631	2631
unique	2	2
top	True	True
freq	1425	2118

```
[4]: data.value_counts()
```

```
[4]: LearnCode  Successful
     True       True          1165
     False      True           953
     True       False          260
     False      False          253
     Name: count, dtype: int64
```

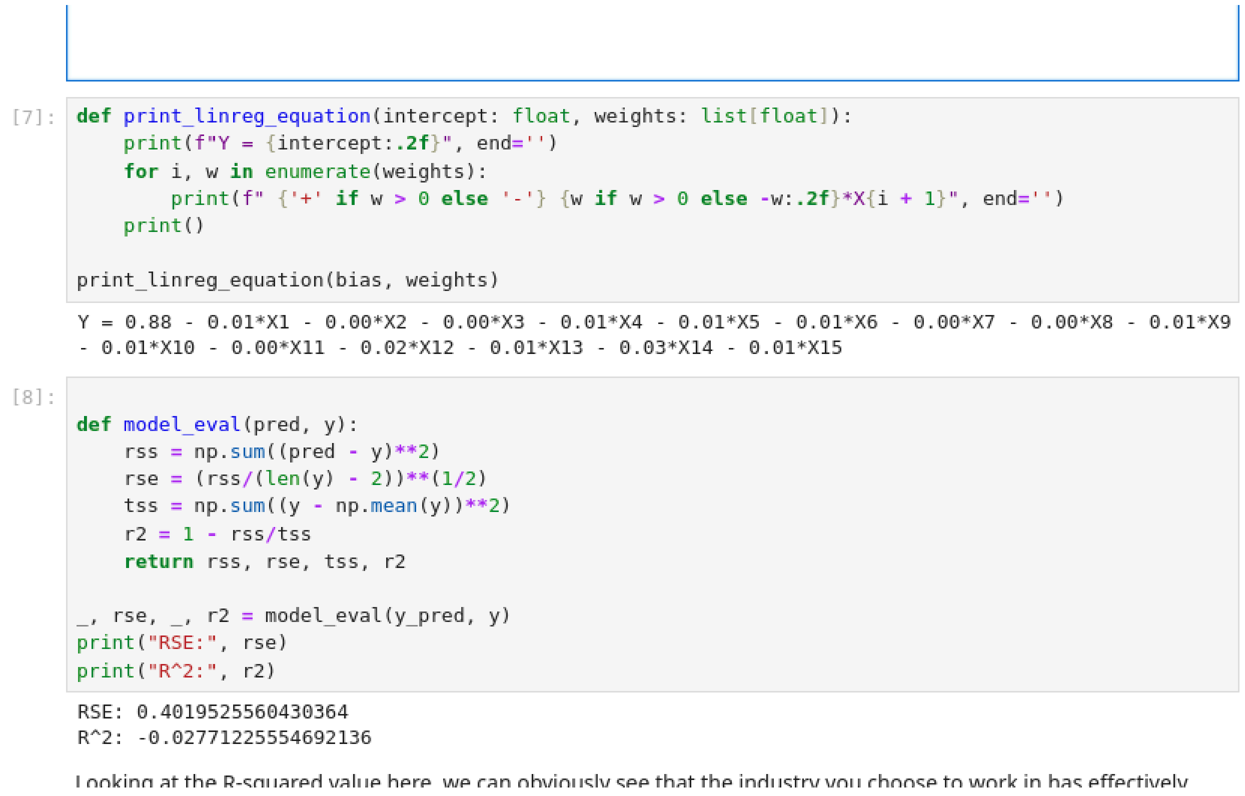
The last of the three misconceptions that I am trying to dispel is unfortunately more complicated than a simple binary predictor and response. I am trying to establish that there is no correlation between the industry chosen to work in and a binary success variable. In this case, I decided to use a simple linear regression model. Now, I know that when you think of categorical response variables, logistic regression or other classifiers are the first choice, but because this is a simple 2-state binary response, linear regression should be accurate. I just want to know how much of an effect, if any, your chosen industry has on your success as a developer. Thanks to one-hot encoding, this is a simple predictor with multiple binary predictors and one binary response.

One other important thing to mention is that the response variable in all of these cases, whether the individual is “successful”, is a rough estimate of success based on whether their job satisfaction is greater than 5/10 and they make enough money to support a family of four (according to the US government).

Results, Conclusions, and Real-World Implications

Looking at the R-squared value here, we can obviously see that the industry you choose to work in has effectively no correlation with your level of success in life. So restricting yourself to specific industries because they are more prestigious or because you think you will be more successful is, in reality, only restricting your possible job opportunities.

I know that the R-squared value being negative shouldn’t happen, but in this case, error accumulated when calculating the statistic because of the high number of very small float multiplications. Regardless, it is clear that the true value is very close to zero because these types of errors are not large.



```
[7]: def print_linreg_equation(intercept: float, weights: list[float]):
         print(f"Y = {intercept:.2f}", end='')
         for i, w in enumerate(weights):
             print(f" {'+' if w > 0 else '-'} {w if w > 0 else -w:.2f}*X{i + 1}", end='')
         print()

     print_linreg_equation(bias, weights)
```

```
Y = 0.88 - 0.01*X1 - 0.00*X2 - 0.00*X3 - 0.01*X4 - 0.01*X5 - 0.01*X6 - 0.00*X7 - 0.00*X8 - 0.01*X9
- 0.01*X10 - 0.00*X11 - 0.02*X12 - 0.01*X13 - 0.03*X14 - 0.01*X15
```

```
[8]:
     def model_eval(pred, y):
         rss = np.sum((pred - y)**2)
         rse = (rss/(len(y) - 2))**(1/2)
         tss = np.sum((y - np.mean(y))**2)
         r2 = 1 - rss/tss
         return rss, rse, tss, r2

     _, rse, _, r2 = model_eval(y_pred, y)
     print("RSE:", rse)
     print("R^2:", r2)
```

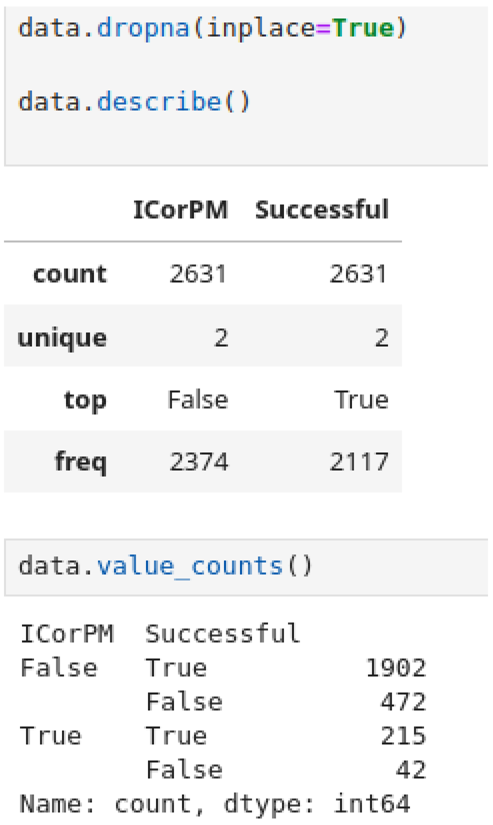
```
RSE: 0.4019525560430364
R^2: -0.02771225554692136
```

Looking at the R-squared value here, we can obviously see that the industry you choose to work in has effectively

Now to analyze our other two results. The data in the screenshots below shows counts of each case, but to calculate the proportions of success in each case, we can just use this simple formula for each case (successful)/(successful + unsuccessful).

For example, the screenshot with LearnCode as the predictor actually describes whether or not the individual learned to code at a college or university or not. This data shows that roughly 80% of software developers are considered "successful" by my very limited classification. More importantly, however, this shows that the proportion is roughly the same, regardless of whether you learned to code in college.

Similarly, in the screenshot with ICorPM as the predictor describes whether or not the person is an individual contributor or a people manager. This once again shows that regardless of whether you are an individual contributor or a people manager, there is still roughly a 80% success ratio for software developers. The ratio for people managers is admittedly slightly higher, at 83.6%, but I'd say that this is within the margin for error with this relatively small number of people managers in this dataset. It could also mean that there is a slight positive association between being a manager and being successful, which would also make sense.

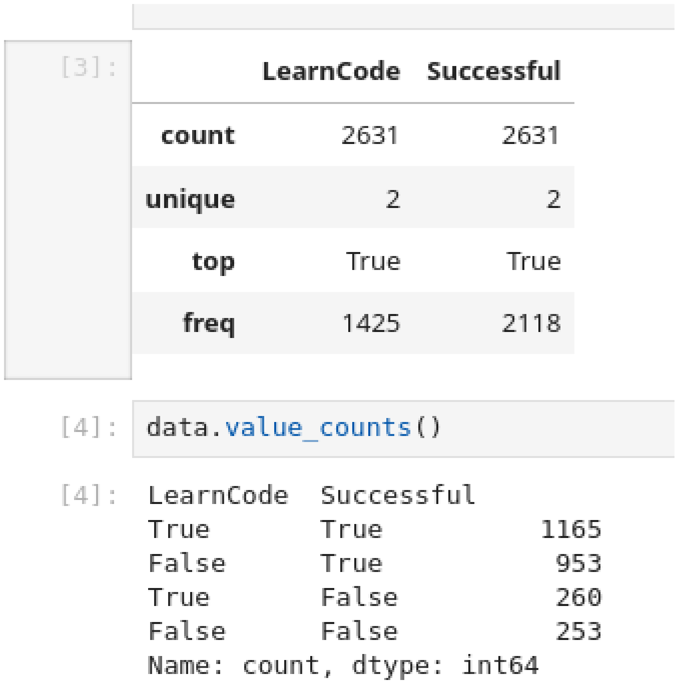


```python
data.dropna(inplace=True)

data.describe()
```

	ICorPM	Successful
count	2631	2631
unique	2	2
top	False	True
freq	2374	2117

```python
data.value_counts()
```

```
ICorPM  Successful
False   True          1902
        False          472
True    True           215
        False           42
Name: count, dtype: int64
```



[3]:

	LearnCode	Successful
count	2631	2631
unique	2	2
top	True	True
freq	1425	2118

```
[4]: data.value_counts()

[4]: LearnCode  Successful
     True       True          1165
     False      True           953
     True       False          260
     False      False          253
     Name: count, dtype: int64
```

In conclusion, these results all show that all of the commonly repeated advice that I wanted to verify objectively are in fact misinformation. According to this dataset and my analysis, a developer can be successful regardless of whether they go to college for computer science, regardless of what industry they end up in, and regardless of whether they end up as individual contributors or managers. Young developers can rest easy knowing that whatever path they take, they have the opportunity to be successful in a career as a software engineer.

I think the big thing this project helped me understand was how to creatively use a dataset to support an argument. I don't mean that in a faking-results sense, I just mean that this dataset isn't nicely set up to allow any easy statistical analysis. I had to do a lot of transformation, interpretation, and encoding, which was a somewhat creative process that I needed to improve at. Most statistics classes give you a perfect dataset that you just need to do the math for, while in this one, I get a lot of freedom in how the data is used. If I ever need/want to work with more less-structured datasets in the future, I feel significantly more capable of turning them into usable data.