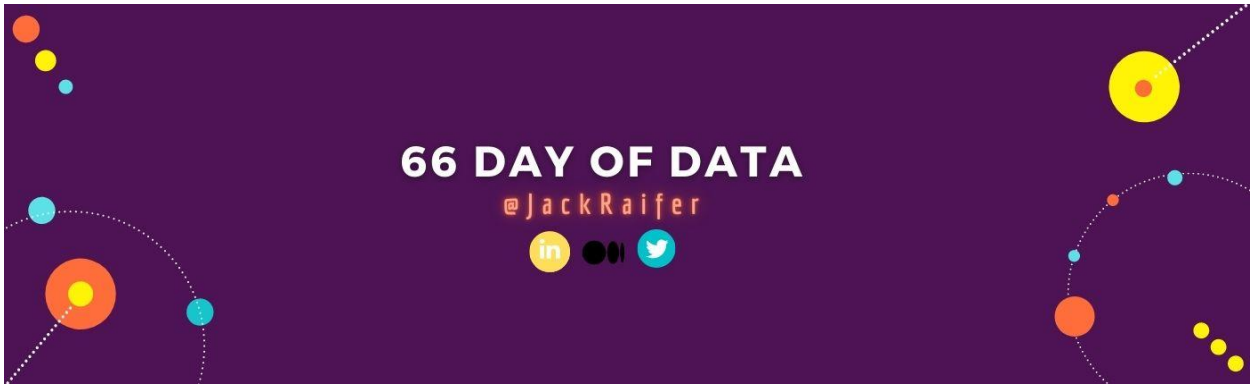

#66DaysOfData – Days 18 to 22: Building a Predictive Model with our Risk Formula

Previously, we created an Attrition Risk formula. Now it is time to build a model with it.

We will keep it simple to start with and build from there in future articles, so it does not become overwhelming. Like always we start by importing the libraries we will need, in this case, NumPy and Pandas will suffice:

```
import numpy as np
import pandas as pd
```

Since this is a formula-based model, it will need some of the static variables we discovered previously, let us create them:

```
[2]: #Formula Variables
     m = [-0.0422, -0.2138, -0.0714, -0.1462,  0.3032] # Variable for the means of correlations
     minres = -0.4736
     maxres = 0.3032
     divisor = 3.74
```

Now that our variables are all set. To do this simply, let us create a function called ***PredAttRisk*** (Predict Attrition Risk). This function will take one single parameter, a list of lists, this is so we can run multiple queries at one time, more on this a bit later. Each list should contain 6 items, first an integer referring to the id number of the entry (or person) and then 5 floating point numbers from 0 to 1 with the data for each of the OCEAN traits in that specific order (Openness, Continuousness, Extraversion, Agreeableness and finally Neuroticism. We will call this list of list ***results*** and it will be the sole parameter for our function. Here is the whole function and I will explain the rest below:

```
# The input data for our function will be an array. More info below.
# The results input should be a list of lists that includes an id number followed by each of the OCEAN results in that order.
def PredAttRisk(results):
    for l in results:
        risk = (((m[0]*l[1] + m[1]*l[2] + m[2]*l[3] + m[3]*l[4] + m[4]*l[5]) - minres) / (maxres - minres)) / divisor
        risk = round(risk, 2)
        print('Risk for ID# {} is {}%.'.format(str(l[0]), risk*100)) #The x100 is to transform the prediction into a percentage
```

As you can see, after stating the function and input, we run a for loop over the **list of lists** results. Then we run the risk formula that we created last time, using the index for each part of the **list** containing our OCEAN information, and combining it with the specific multiplier from our **means list** above, and then we plug in the rest of the **variables** to complete our formula.

Then we add an extra line to round our result to 2 decimal places (if we don´t do this, our results will get unwieldly).

Finally, we print the result, in a nice way using the **.formta** method. Just a note, in biostatistics, risk is calculated on normalized basis (between 0 and 1), so to make it more readable, we multiply the result by 100.

Let us create a test variable with only one list for our input:

```
[33]: # List for singular test.
      test1 = [[1, 0.803, 0.886, 0.496, 0.753, 0.426]]
```

Do note the double square brackets, since our input takes in a list of lists, in this case it is a list with a single list in it. Now we run our function with **test1** as input and see what happens:

```
34]: # Testing with 1 list
     PredAttRisk(test1)

     Risk for ID# 1 is 8.0%.
```

And TA-DA, it works. This first result is predicted to have an 8% risk of attrition. Now we know it works for one entry, let us now try it with several ones. For that, we will create a list of lists with 5 lists:

```
[35]: # Now let us try it with a few more lists at the same time
      test2 = [[1, 0.803, 0.886, 0.496, 0.753, 0.426],
               [2, 0.503, 0.766, 0.855, 0.621, 0.519],
               [3, 0.731, 0.432, 0.631, 0.859, 0.622],
               [4, 0.600, 0.616, 0.716, 0.636, 0.563],
               [5, 0.462, 0.368, 0.425, 0.526, 0.942],
              ]
```

And drum roll please:

```
[36]: PredAttRisk(test2)

Risk for ID# 1 is 8.0%.
Risk for ID# 2 is 10.0%.
Risk for ID# 3 is 13.0%.
Risk for ID# 4 is 12.0%.
Risk for ID# 5 is 19.0%.
```

Seems to be working well. But we can do one more test, which is using data straight from a data frame (like the ones we found a while back with OCEAN data) and running the function to make sure it is working well. For this, we first reload the data from the CSV file into the data frame df1:

```
]: #Testing with a dataframe
df1 = pd.read_csv('DATA_OCEAN/big_five_scores.csv')
df1.head()
```

]:

	case_id	country	age	sex	agreeable_score	extraversion_score	openness_score	conscientiousness_score	neuroticism_score
0	1	South Afri	24	1	0.753333	0.496667	0.803333	0.886667	0.426667
1	3	UK	24	2	0.733333	0.680000	0.786667	0.746667	0.590000
2	4	USA	36	2	0.880000	0.770000	0.860000	0.896667	0.296667
3	5	UK	19	1	0.690000	0.616667	0.716667	0.636667	0.563333
4	6	UK	17	1	0.600000	0.713333	0.646667	0.633333	0.513333

Then we create a test3 which will be a list of lists created directly from our data frame. So as not to get overwhelmed with the results, since right now they get printed to the console, we will use the **.head()** method to limit the results to 15:

```
4]: test3 = []
for ind in df1.head(15).index:
    test3.append([df1['case_id'][ind],
                  df1['openness_score'][ind],
                  df1['conscientiousness_score'][ind],
                  df1['extraversion_score'][ind],
                  df1['agreeable_score'][ind],
                  df1['neuroticism_score'][ind]])
```

And finally, we run our function on the **test3** variable and cross our fingers (it did work, but not first without having to fix some typos, you know, usual programing stuff):

```
[78]: PredAttRisk(test3)

Risk for ID# 1 is 8.0%.
Risk for ID# 3 is 10.0%.
Risk for ID# 4 is 5.0%.
Risk for ID# 5 is 11.0%.
Risk for ID# 6 is 11.0%.
Risk for ID# 7 is 13.0%.
Risk for ID# 8 is 9.0%.
Risk for ID# 9 is 10.0%.
Risk for ID# 10 is 13.0%.
Risk for ID# 11 is 12.0%.
Risk for ID# 12 is 11.0%.
Risk for ID# 13 is 4.0%.
Risk for ID# 14 is 12.0%.
Risk for ID# 15 is 11.0%.
Risk for ID# 16 is 16.0%.
```

And there you have it we have created a basic risk prediction model that outputs the risk of attrition / turnover based on OCEAN personality results. But, as always, you need to think of how this would be used in real life. Most professionals would consider a 16% risk to be exceptionally low, when in fact, it is remarkably high being our average 10.9%. This means that next time, we need to make some tweaks to our model so that it is more readable towards an end user.

See you then…

And remember you can access all the datasets, notebooks and everything related to this project here on my github repo.

Next Time – Tweaking our model towards usability.

Jack Raifer Baruch

Follow me on Twitter: @JackRaifer

Follow me on LinkedIN: jackraifer

About the Road to Data Science - #66DaysOfData Series

Road to Data Science series began after I experienced the first round of Ken Jee´s #66DaysOfData challenge back in 2020. Since we are starting the second round of the challenge, I thought it would be a good idea to add small articles every day where I can comment my progress.

I will be sharing all the notebooks, articles and data I can on GitHub: https://github.com/jackraifer/66DaysOfData-Road-to-Data-Science

Please do understand I might have to withhold some information, including code, data, visualizations and/or models, because of confidentiality regards. But I will try to share as much as possible.

Want to follow the #66DaysOfDataChallenge?

Just follow Ken Jee on twitter **@KenJee_DS** and join the #66DaysOfData challenge.

You can also reach out to me at any time through **LinkedIN** or **Twitter.**