CSCI 720 Big Data Analytics HW03 Results

Student: Guo, Zizhun
Email: zg2808@cs.rit.edu
Phone: 585-284-0464
Submission: Feb/15/2020
Due Date: Feb/16/2020 11:59 PM

---

1. Did your mentor program use pandas to help you along?

Yes.

```
    # location: HW_03_Guo_Zizhun_Mentor.py

    import pandas as pd # data analysis and manipulation package

```

2. Did your mentor program create your training program?

Yes.

```
    # location: HW_03_Guo_Zizhun_Mentor.py

    def write_file(best_attribute, is_positive_correlated):
    """ writes trained program based on selected best attribute
    Paras:
        @best_attribute: a string represents the name of best attribute
        @is_positive_correlated: a boolean determines the classification rule
    Return:
        void
    """
```

```
# initialize a string to contains codes for dumping in the trained program
    lines = ""
        ..
        .. implementations ..
        ..
    # create a file with given name "filepath"
    f = open('HW03_Guo_Zizhun_Trained.py', "w")
    # write string to the filepath
    f.write(lines)
    f.close()
```

3. Did the training program use pandas?

Yes.

```
    # location: HW_03_Guo_Zizhun_Trained.py

    import pandas as pd

```

4. Which attribute is most strongly cross-correlated with the target variable?

Attribute	Correlation with Target variable
Bread	-0.012
Vitamins	-0.448
Vegetable	0.016
Milk	-0.035
Banana	-0.070
PeanutButter	0.582
Chocolate	-0.030
Citrus	-0.057
Cookie	0.050
IceCream	0.013
Soda	-0.061
Fruit	-0.005

PeanutButter has the highest cross correlation absolute value, which means it is the most strongly cross-correlated attribute with the target variable.

5. Which is best feature out of the above items for your One Rule?

PeanutButter.

6. What structure did your One Rule classifier have?

The if-statement condition sets the strucure based on whether the **value of cross-correlation** is **positive** or **negative**.
Sets structure for One Rule classifier in trained program:

```
    # location: HW_03_Guo_Zizhun_Mentor.py
    if is_positive_correlated:

        lines += f"\n\tfor val in data:"
        lines += f"\n\t\tif val >  0:"
        lines += f"\n\t\t\tprint('1')"
        lines += f"\n\t\telse:"
        lines +=  f"\n\t\t\tprint('0')"
    else:
        lines += f"\n\tfor val in data:"
        lines += f"\n\t\tif val ==  0:"
        lines += f"\n\t\t\tprint('1')"
        lines += f"\n\t\telse:"
        lines +=  f"\n\t\t\tprint('0')"

```

Q1: What was the if-else rule you got?

Struture below: (based on the given training dataset)

```
    # location: HW_03_Guo_Zizhun_Trained.py

    if attribute >  0:
        print('1')
    else:
        print('0')

```

7. Run the original training data back through your classifier.

Q1: What was the accuracy of your resulting classifier, on the training data?

The Trained Program would print the result as the homework asked, but the accuracy can be captured by **frequency table** created in Mentor Program:

```
# shell console

PeanutButter  Sickness
0             0           399
              1           108
1             0           101
              1           392
dtype: int64

```

$numberOfCorrectness = numberOf(PeanutButter:0, Sickness:0) + numberOf(PeanutButter:1, Sickness:1) = 399 + 392 = 791$
$numberOfTotal = 1000$
$Accuracy = numberOfCorrectness/numberOfTotal = 791/1000 = 0.79$

The **Accuracy** is **0.79**.

*Reference:
Another way to define Accuracy: $ACC = (TP + TN)/(TP + TN + FP + FN)$

8. Conclusion: (2)

Q1: What else did you learn along the way here?

I also leanred using scatter plot to present the correlation situation.
The default image would be looks like this:

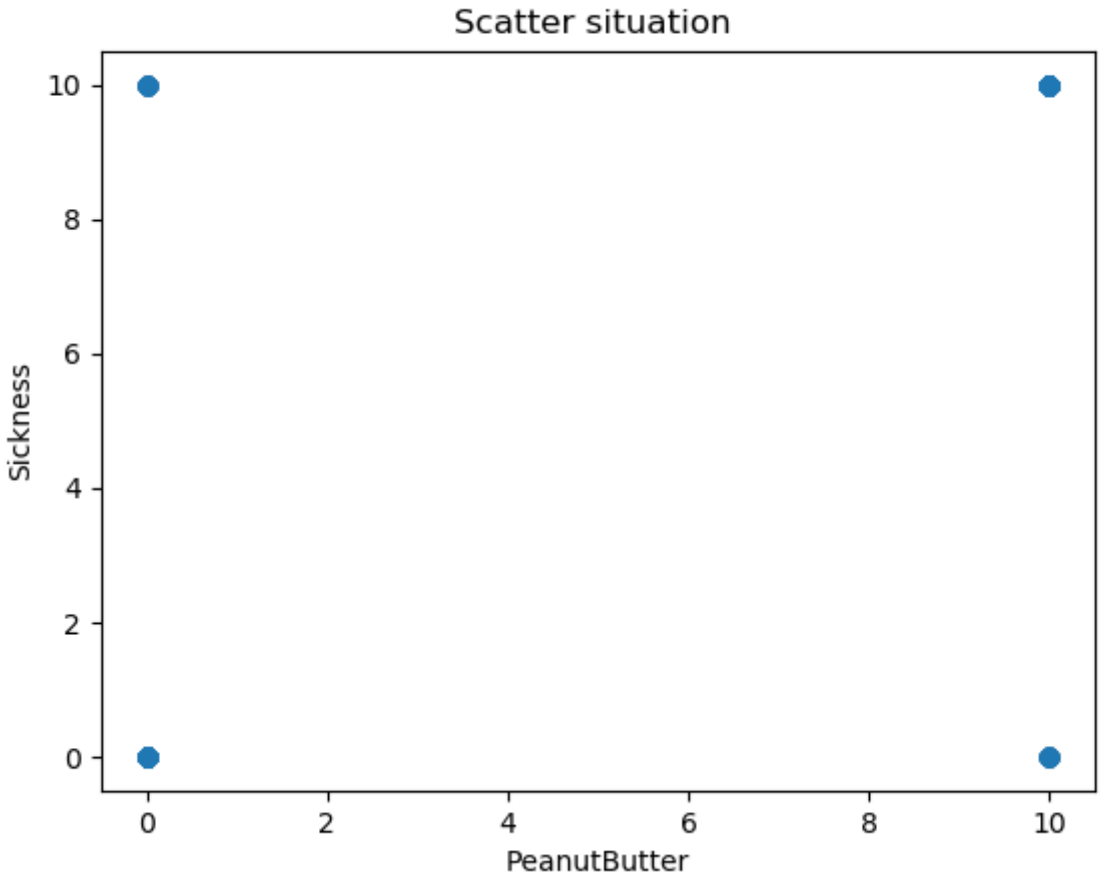

Image 1: The 2D Sickness based on Peanutbutter

Each dot actually overlaps so many points. In order to see how condense each points are, we need to add jitter to scatter it up a little bit. So just choose a randomly scattered seed generated by Numpy package.

```
    # location: HW_03_Guo_Zizhun_Mentor/ditribution_plot_with_jitter.py

    x = df_filtered[best_attribute] \
                * (1 - scatter_fraction_rate) \
                * scatter_scale \
            + np.random.ranf(size) \
                * scatter_fraction_rate \
                * scatter_scale
    ...
    ...

    plt.scatter(x, y, alpha=0.5)    # plot the image alpha = 0.5 indicates the level of
                                    # Area with darker color have more points overlappe
```

After adding the **Jitter**:

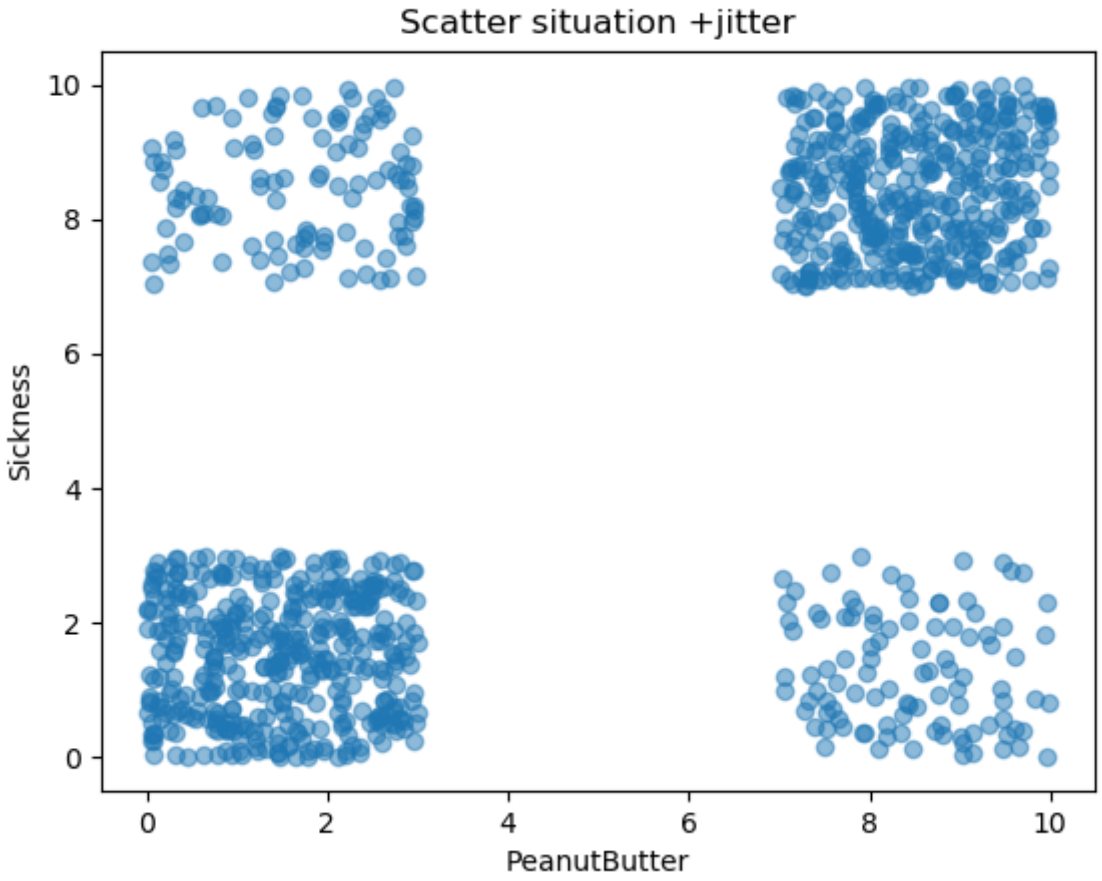

Image 2: The 2D Sickness based on Peanutbutter with Jitter

Here **Jitter** as the **Noise** added enabling easy observation on how the points are settles. As can be seen, points from **lower left** and **upper right** are largely placed, which means the two x-variable and y-variable are **strongly positive cross-correlated**.

This tendency implies: The y-variable tends to have the same value of x_variable, since in our case, we only has two value options (0 and 1).

Q2: What can you conclude?

The tendency for attribute values to be changed as the target vairables is defined by the **sign** of **cross-correlation**. This conclusion can be used to determine the One Rule of trained programe that if the cross-correlation(CC) is less than 0, the prediction should be **opposite** as the value of attribute, whereas if CC is greater than 0, the prediction should be **same** result of the attribute value.

9. BONUS: (+1)

Q1: In addition to the best feature, can you find another feature that might help classify sickness?

Yes. **Vitamin** can be this feature, since

Q2: What makes you say this?

Because it has the second highest cross-correlation, which is **-0.448**.

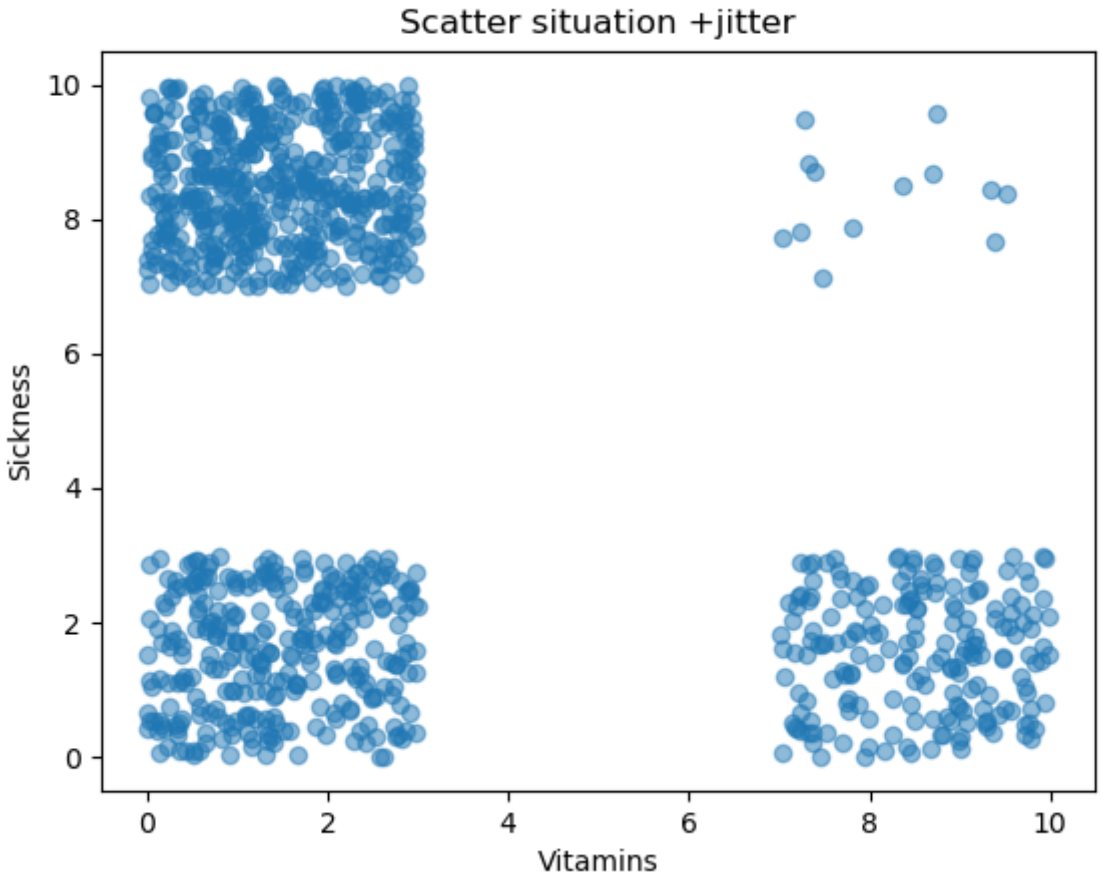

Image 3: The 2D Sickness based on Vitamin with Jitter

Q3: How did you test this hypothesis?

I can print the frequency table to calculate the Accuracy.

Vitamins	Sickness	#
0	0	305
	1	487
1	0	195
	1	13
		total: 1000

Accuracy = (487+195)/1000 = **0.682**

*Since the correlation is less than 0, it needs to sum up the # of {0, 1} and {1, 0} pairs.