Sensor Simulations Challenge *by Kaimin (Kelvin) Deng*

- **Email**: k4tang@gmail.com
- **Phone**: 647-885-5944

Question 1

"red sky in the morning, sailor take warning"

From the definition, **Independent Event** means the outcome of one event does not affect the outcome of the other:

$$P(A \cap B) = P(A) \cdot P(B)$$

Where event **A** is "red sky" and event **B** is "storm".

$$P(A) = (10 + 40)/(10 + 40 + X + 60) = 50/(110 + X)$$
$$P(B) = (10 + X)/(10 + 40 + X + 60) = (10 + X)/(110 + X)$$
$$P(A \cap B) = 10/(10 + 40 + X + 60) = 10/(110 + X)$$

Then $X = 15$

The Pearson's **Chi-Squared Test** can be used to determine if the storm weather is independent from the red sky.

```
import numpy as np
from scipy.stats import chi2_contingency

# Assuming the X is 5, then
obs = np.array([[10, 5], [40, 60]])
stat, p, dof, expected = chi2_contingency(obs)
print('The test statistic: {:0.4f}'.format(stat))
print('The p-value: {:0.4f}'.format(p))
print('The degree of freedom: {:0.4f}'.format(dof))
print('The expected value: {}'.format(expected))
```

```
The test statistic: 2.7672
The p-value: 0.0962
The degree of freedom: 1.0000
The expected value: [[ 6.52173913  8.47826087]
 [43.47826087 56.52173913]]
```

H_0 hypothesis: Storm is indenpendent to red sky.

H_1 hypothesis: Storm is dependent to red sky.

```
# If alpha is given as 0.05
alpha = 0.05
if p <= alpha:
  print('Reject H0 hypothesis. Dependent')
else:
  print('Fail to reject H0 hypothesis. Independent')
```

```
Fail to reject H0 hypothesis. Independent
```

Question 2: Testing Apple tree group differences

2.1 Read file

[ ] ↳ *2 cells hidden*

2.2 Four groups of samples

```
df = df.apply(pd.to_numeric)
df.head()
```

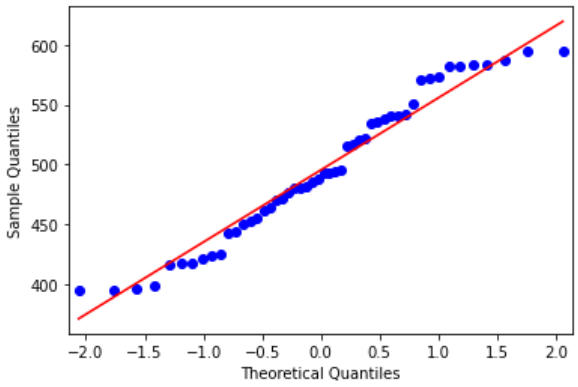
	S1_treated	S1_untreated	S2_treated	S2_untreated
0	488.0	458.0	455.0	436.0
1	541.0	583.0	435.0	512.0
2	494.0	478.0	543.0	523.0
3	536.0	562.0	469.0	444.0
4	417.0	577.0	503.0	394.0

2.3 Normal Distribution & Homogeneity of Variance

First of all, we should check if all samples are distributed normally. A **Q-Q plot (Quantile-Quantile Plot)** was used to verify the distribution of samples is close to the normal distribution.

```
from statsmodels.graphics.gofplots import qqplot
from matplotlib import pyplot
# q-q plot
qqplot(df.S1_treated, line='s')
pyplot.show()
```

```
/usr/local/lib/python3.7/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecat
  import pandas.util.testing as tm
```



Let's check if two sample groups have a similar amount of variance (i.e. Homogeneity of Variance).

```
from scipy import stats
# Season 1
print(stats.levene(df.S1_treated, df.S1_untreated))

# Season 2
print(stats.levene(df.S2_treated, df.S2_untreated))
```

```
LeveneResult(statistic=0.004267968118125966, pvalue=0.9480445889021037)
LeveneResult(statistic=2.658904987818953, pvalue=0.10618150115782979)
```

Through the Levene test result, Season 1 group has a p-value $(0.948) >> 0.05$, and the Season 2 groupd has a p-value $(0.106) > 0.05$. It means it fails to reject the null hypothesis. We can consider both samples have equal variance.

Finally, two groups have a similar amount of variance, we should use keep `equal_var` as default value `True`.

```
(s1_statistic, s1_pvalue) = stats.ttest_ind(df.S1_treated, df.S1_untreated)
print("statistic: {:.4f}".format(s1_statistic))
print("p value: {:.4f}".format(s1_pvalue))
```

```
statistic: 0.7244
p value: 0.4706
```

```
(s2_statistic, s2_pvalue) = stats.ttest_ind(df.S2_treated, df.S2_untreated)
print("statistic: {:.4f}".format(s2_statistic))
print("p value: {:.4f}".format(s2_pvalue))
```

```
statistic: 3.1148
p value: 0.0024
```

2.4 Conclusion

When setting the significant level $\alpha = 0.05$, the treatments for Season 1 and Season 2 are behaving differently.

From the p-value of Season 1, we can see $0.4706 >> \alpha$. We fail to reject the null hypothesis H_0 (It has no statistical significance between two samples).

While the p-value of Season 2 is much smaller than α. We have very high confidence to reject the null hypothesis H_0.

If we set the significant level $\alpha = 0.01$, and the p-value is still smaller than it. We would have **99% of confidence** that the Season 2 product has a significant difference from the untreated group.

Question 3: Flow through a hose

3.1 Read file

[ ] ↳ *2 cells hidden*

3.2 Define function for SVM model

[ ] ↳ *1 cell hidden*

3.3 SVM Model Result

[ ] ↳ *1 cell hidden*

3.4 Discussion

As the result of the error rates above, they all have a **0% error rate**, which means SVM and 100% predict whether the hose cross-section is too small (*Bad* region) or big enough (*Acceptable* region).

This is impossible in a real-life situation. Although simulating the hose flow and RPM based on physical equation **spends less time than measuring these parameters physically in the cars**, the **generated data cannot represent the actual situation of the car**. It is

noteworthy that the actual data consists of many kinds of "noise" which are very difficult to reproduce through a simulation experiment like this.

Given the pros and cons of simulation and physical analysis, a combination of both methods would save time in modelling and be

Question 4: A simple car suspension model

For the detailed codes and explanation (Confusion Matrix, Accuracy and Error Rate), please refer to the 2nd notebook.

4.1 Result Table

Here is the **Result Table** of 3 models with different parameters (4 replications)

ML algorithm	N runs	experiment type	observation type	error rates	result
KNN	100	random roads	road input	0.03400	success
KNN	200	random roads	road input	0.02800	success
KNN	200	one standard road	road input	0.00000	success
KNN	200	random roads	in vehicle vibration	0.04550	success
KNN	200	one standard road	in vehicle vibration	0.00000	success
KNN	500	random roads	road input	0.01910	success
Naive Bayes	200	random roads	road input	0.34888	fail
Naive Bayes	200	one standard road	in vehicle vibration	0.00000	success
Naive Bayes	200	random roads	in vehicle vibration	0.15037	success
Naive Bayes*	200	one standard road	road input	0.31225	fail
Naive Bayes	1000	random roads	road input	0.34304	fail
MLP(20,5)	200	random roads	road input	0.00605	success
MLP(20,5)	500	random roads	road input	0.00584	success
MLP(20,5)	1000	random roads	road input	0.00419	success
MLP(20,5)	500	one standard road	road input	0.00024	success
MLP(20,5)	500	one standard road	in vehicle vibration	0.00000	success
MLP(20,5)	500	random roads	in vehicle vibration	0.03952	success

With the 20% threshold of an error rate, only the **Naive Bayes** model got failed results.

1. K-Nearest Neighbour (KNN):

KNN is a non-parametric algorithm. So it does not make any assumption on the data distribution. The main parameter it uses is the number of nearest neighbours. Objects are classified based on the most common class among the neighbours by their spacial distances (Similarity).

In other words, **it has the advantage of easy implementation and it does not need to train some parameters**. Very effective to **noisy data (with random noise inputs from roads)**, and a big dataset (in this case we have 10k, 20k, 50k, 100k rows).

However, it has the disadvantages that the k value needs to be determined by a human, and the computation cost would get much higher as the number of features gets higher.

2. Naive Bayes Classifier

This algorithm is naive because it assumes simply that every attribute is conditionally independent of each other.

$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

This simple assumption results in an extremely fast computation. **And it has the advantage of scalability when it comes to a large dataset, while other complicated models might take much more time exponentially**.

However, Naive Bayes's simplicity leads to bad predictions. In this case, it is the only model that failed the requirement (< 20%). As it is shown on the table, when the observation type is not `in vehicle vibration`, the error rate will increase to over 30% regardless of the experiment type.

PS: A new case was added (see the * line, **one standard road|road input**), and the result still failed. On the contrary, the KNN and MLP models got a very good accuracy in this case.

3. Multilayer Perceptron (MLP)

As a basic form of artificial neural network (ANN), MLP can get a great result for classification problems.

Neural networks are good to **model with nonlinear large data**. From the table, the error rate decreases from *0.00605 to 0.00419* when the number of runs increases from 200 to 1000. Although the training part takes a longer time as it needs to "learn" from the previous result and improves the weights for inputs, **it has the best result among these three models**. It also **predicts pretty fast** once it is trained.

But neural networks **can only take numeric inputs**. Therefore, we need to transform the **label (y_rn) into binary values**. And it is hard to tune the model since the layers **are black boxes**. In addition, it requires a lot of guesswork since we **cannot see directly how much each independent variable is influencing others**.

4.2 Case Comparison

From the result table, all models got a relatively high error rates in the case of **random roads** and **road input**. While the case of **standard road** and **in vehicle vibration** gave the lowest error rates among all combinations.

1. standard road & in vehicle vibration (easiest)

In the easiest case, there is **no random variable from the road and car damper**, which makes the model much easier to predict the car failure. As can be seen from the result, all three models **give 0.0% error rate** even for the worst model Naive Bayes.

2. random roads & in vehicle vibration

Once the **random road experiment** is introduced in the model, the result gets worse significantly since it has **one random variable from the road**. The model becomes a bit harder to determine whether the failure comes from the road condition or not.

3. one standard road & road input

When the road input is used, the result gets much worse compared to the second case. This introduces a variable to the model. **The vertical forces are generated by both the road and mass position response**. This results in the "noise" to the model, giving a higher error rate.

4. random roads and road input (hardest)

Combining the second and third cases, this case **introduces two variables to the model at the same time**, making the model much hard to predict the reason of failure.

In conclusion, the more variables, the harder the prediction. But more runs (larger dataset) can lower the error rate to get a better model prediction of car failure. Besides, MLP gives the best result among them.