Need help with Python Machine Learning? Take the FREE Mini-Course

How to Use Statistical Significance Tests to Interpret Machine Learning Results

by **Jason Brownlee** on May 3, 2017 in **Python Machine Learning**

It is good practice to gather a population of results when comparing two different machine learning algorithms or when comparing the same algorithm with different configurations.

Repeating each experimental run 30 or more times gives you a population of results from which you can calculate the mean expected performance, given the stochastic nature of most machine learning algorithms.

If the mean expected performance from two algorithms or configurations are different, how do you know that the difference is significant, and how significant?

Get Your Start in Machine Learning

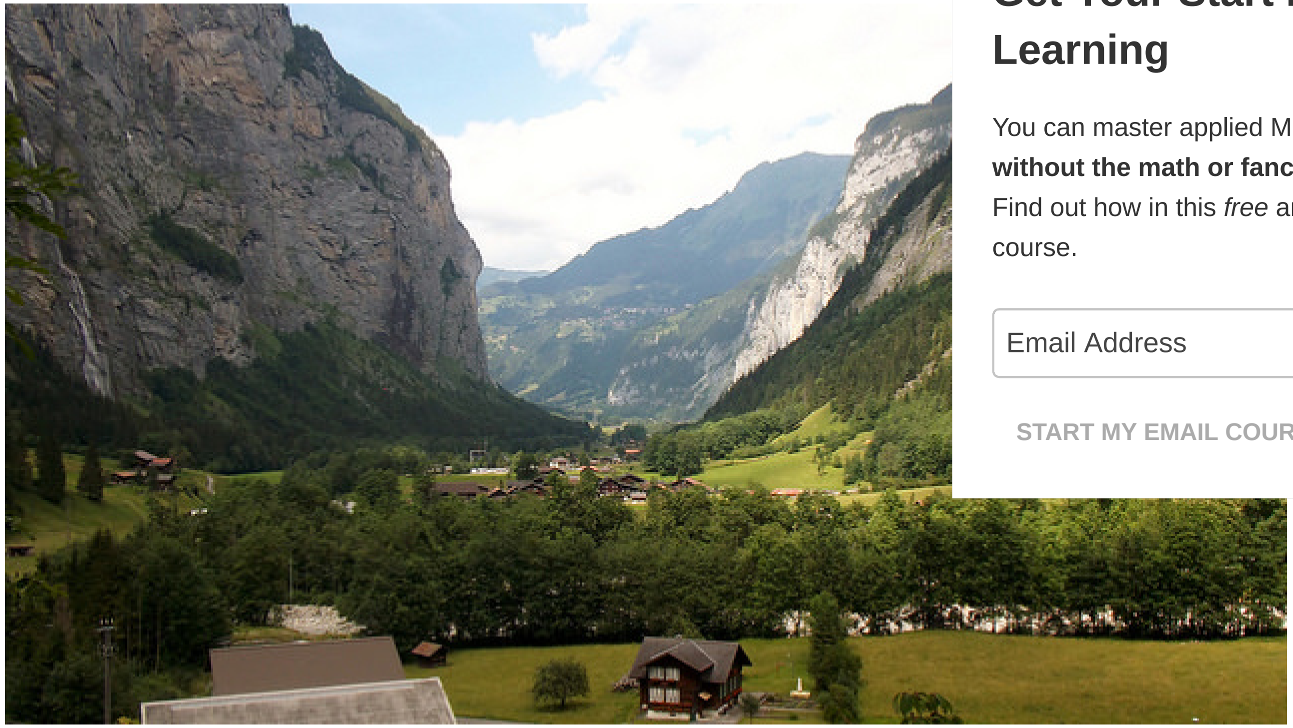
Statistical significance tests are an important tool to help to interpret the results from machine learni[redacted]se tools can help you better and more confidently present your experimental results and choose the right algorithms and configurations for your predictive modeling problem.

In this tutorial, you will discover how you can investigate and interpret machine learning experimental results using statistical significance tests in Python.

After completing this tutorial, you will know:

- How to apply normality tests to confirm that your data is (or is not) normally distributed.
- How to apply parametric statistical significance tests for normally distributed results.
- How to apply nonparametric statistical significance tests for more complex distributions of results.

Let's get started.


How to Use Statistical Significance Tests to Interpret Machine Learning Results
Photo by oatsy40, some rights reserved.

Tutorial Overview

This tutorial is broken down into 6 parts. They are:

1. Generate Sample Data
2. Summary Statistics
3. Normality Test
4. Compare Means for Gaussian Results
5. Compare Means for Gaussian Results with Different Variance
6. Compare Means for Non-Gaussian Results

This tutorial assumes Python 2 or 3 and a SciPy environment with NumPy, Pandas, and Matplotlib.

Generate Sample Data

The situation is that you have experimental results from two algorithms or two different configuration

Each algorithm has been trialed multiple times on the test dataset and a skill score has been collecte

We can simulate this by generating two populations of Gaussian random numbers distributed aroun

The code below generates the results from the first algorithm. A total of 1000 results are stored in a m a Gaussian distribution with the mean of 50 and the standard deviation of 10.

```python
from numpy.random import seed
from numpy.random import normal
from numpy import savetxt
# define underlying distribution of results
mean = 50
stev = 10
# generate samples from ideal distribution
seed(1)
results = normal(mean, stev, 1000)
# save to ASCII file
savetxt('results1.csv', results)
```

Below is a snippet of the first 5 rows of data from *results1.csv*.

```
6.624345363663240960e+01
4.388243586349924641e+01
4.471828247736544171e+01
3.927031377843829318e+01
5.865407629324678851e+01
...
```

We can now generate the results for the second algorithm. We will use the same method and draw the results from a slightly different Gaussian distribution (mean of 60 with the same standard deviation). Results are written to *results2.csv*.

```python
from numpy.random import seed
from numpy.random import normal
from numpy import savetxt
# define underlying distribution of results
mean = 60
stev = 10
# generate samples from ideal distribution
seed(1)
results = normal(mean, stev, 1000)
# save to ASCII file
savetxt('results2.csv', results)
```

Below is a sample of the first 5 rows from *results2.csv*.

```
7.624345363663240960e+01
5.388243586349924641e+01
5.471828247736544171e+01
4.927031377843829318e+01
6.865407629324678851e+01
...
```

Going forward, we will pretend that we don't know the underlying distribution of either set of results.

I chose populations of 1000 results per experiment arbitrarily. It is more realistic to use populations of 30 or 100 results to achieve a suitably good estimate (e.g. low standard error).

Don't worry if your results are not Gaussian; we will look at how the methods break down for non-Gaussian data and what alternate methods to use instead.

Summary Statistics

The first step after collecting results is to review some summary statistics and learn more about the [redacted]

This includes reviewing summary statistics and plots of the data.

Below is a complete code listing to review some summary statistics for the two sets of results.

```
from pandas import DataFrame
from pandas import read_csv
from matplotlib import pyplot
# load results file
results = DataFrame()
results['A'] = read_csv('results1.csv', header=None).values[:, 0]
results['B'] = read_csv('results2.csv', header=None).values[:, 0]
# descriptive stats
print(results.describe())
# box and whisker plot
results.boxplot()
pyplot.show()
# histogram
results.hist()
pyplot.show()
```

The example loads both sets of results and starts off by printing summary statistics. Data in *results1.* "B" for brevity.

We will assume that the data represents an error score on a test dataset and that minimizing the sco

We can see that on average A (50.388125) was better than B (60.388125). We can also see the sar t the standard deviations, we can also see that it appears both distributions have a similar (identical) s

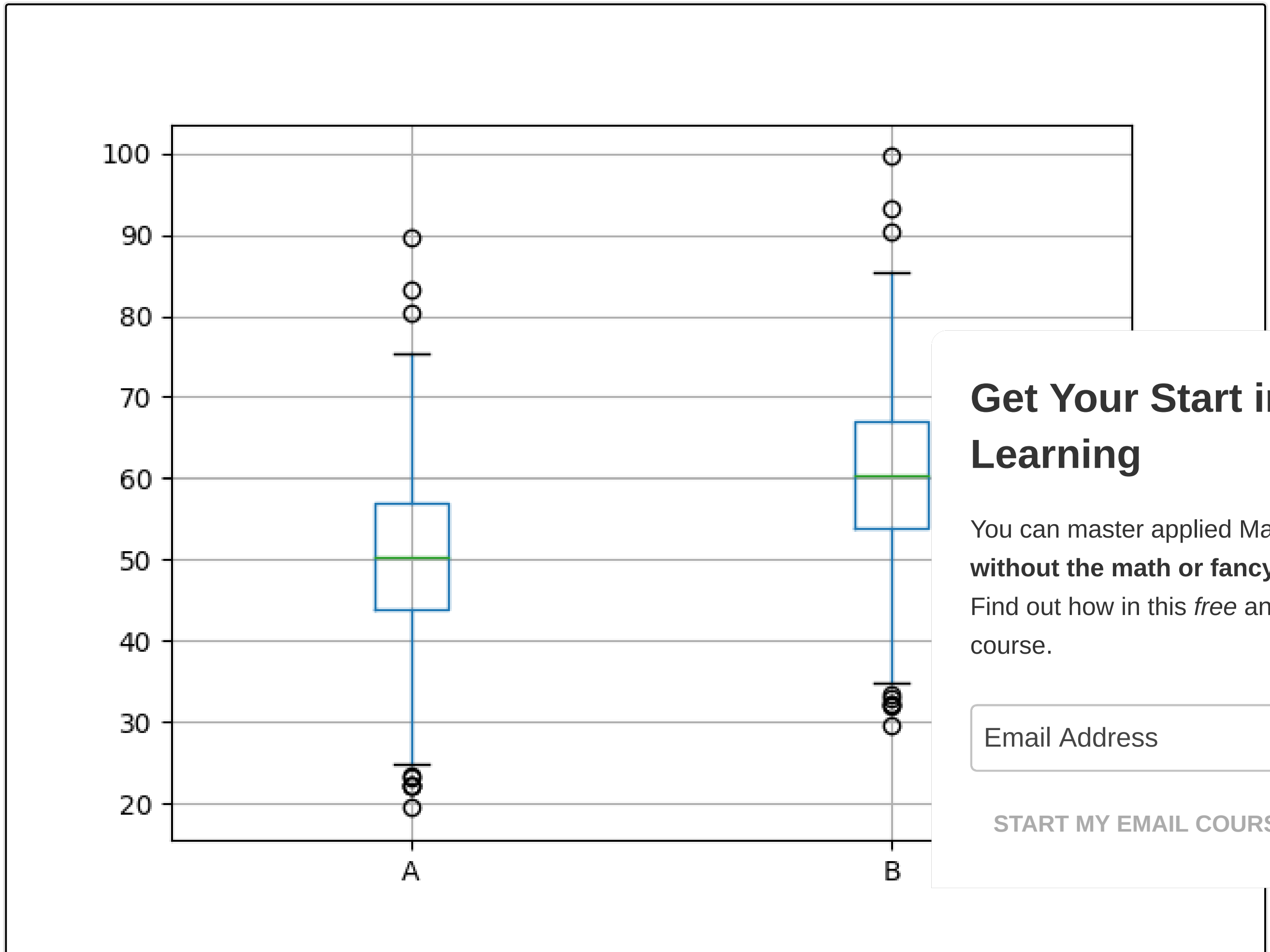
```
                 A            B
count  1000.000000  1000.000000
mean     50.388125    60.388125
std       9.814950     9.814950
min      19.462356    29.462356
25%      43.998396    53.998396
50%      50.412926    60.412926
75%      57.039989    67.039989
max      89.586027    99.586027
```

Next, a box and whisker plot is created comparing both sets of results. The box captures the middle he green line shows the median. We can see the data indeed has a similar spread from both distributio

[redacted]

median.

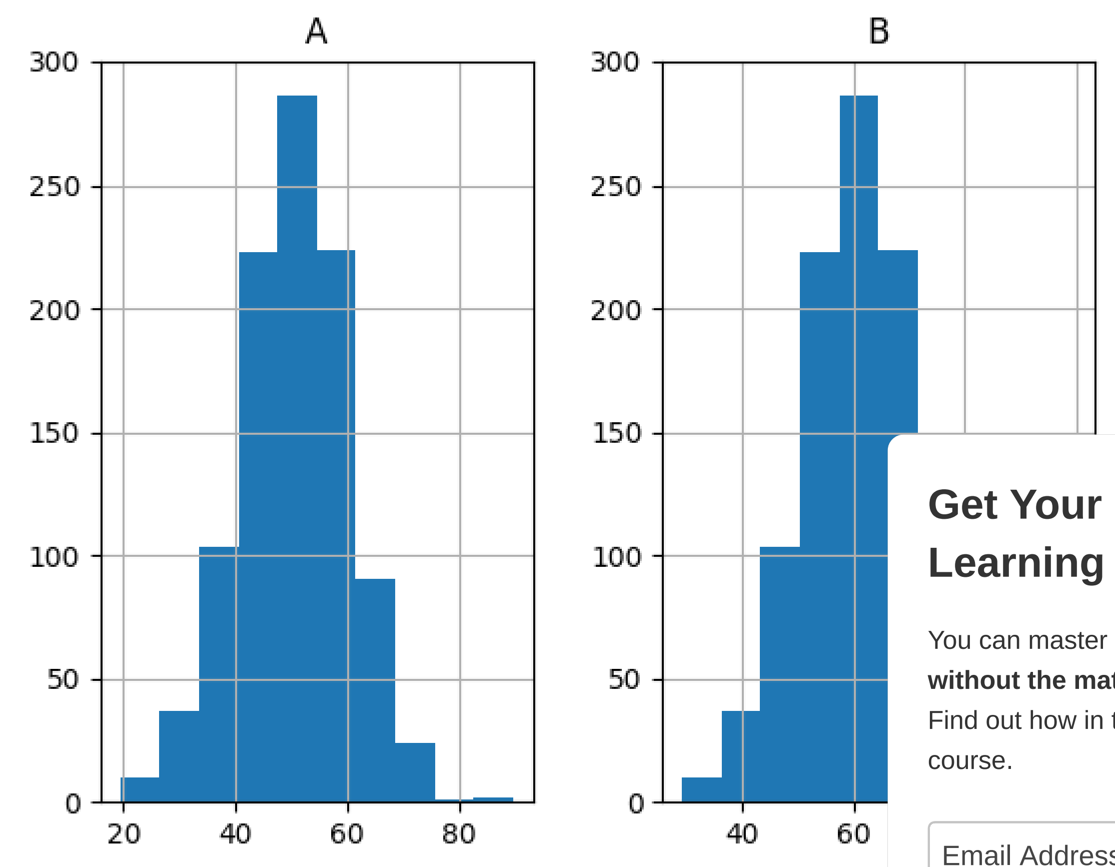
The results for A look better than B.



Box and Whisker Plots of Both Sets of Results

Finally, histograms of both sets of results are plotted.

The plots strongly suggest that both sets of results are drawn from a Gaussian distribution.



Histogram Plots of Both Sets of Results

Normality Test

Data drawn from a Gaussian distribution can be easier to work with as there are many tools and techniques specifically designed for this case.

We can use a statistical test to confirm that the results drawn from both distributions are Gaussian (also called the normal distribution).

In SciPy, this is the normaltest() function.

From the documentation, the test is described as:

[redacted]

> *Tests whether a sample differs from a normal distribution.*

The null hypothesis of the test (H0), or the default expectation, is that the statistic describes a normal distribution.

We accept this hypothesis if the p-value is greater than 0.05. We reject this hypothesis if the p-value <= 0.05. In this case, we would believe the distribution is not normal with 95% confidence.

The code below loads *results1.csv* and determines whether it is likely that the data is Gaussian.

```python
from pandas import read_csv
from scipy.stats import normaltest
from matplotlib import pyplot
result1 = read_csv('results1.csv', header=None)
value, p = normaltest(result1.values[:,0])
print(value, p)
if p >= 0.05:
    print('It is likely that result1 is normal')
else:
    print('It is unlikely that result1 is normal')
```

Running the example first prints the calculated statistic and the p-value that the statistic was calculat

We can see that it is very likely that *results1.csv* is Gaussian.

```
2.99013078116 0.224233941463
It is likely that result1 is normal
```

We can repeat this same test with data from *results2.csv*.

The complete code listing is provided below.

```python
from pandas import read_csv
from scipy.stats import normaltest
from matplotlib import pyplot
result2 = read_csv('results2.csv', header=None)
value, p = normaltest(result2.values[:,0])
print(value, p)
if p >= 0.05:
    print('It is likely that result2 is normal')
else:
    print('It is unlikely that result2 is normal')
```

Running the example provides the same statistic p-value and outcome.

Both sets of results are Gaussian.

```
2.99013078116 0.224233941463
It is likely that result2 is normal
```

Compare Means for Gaussian Results

Both sets of results are Gaussian and have the same variance; this means we can use the Student t-test to see if the difference between the means of the two distributions is statistically significant or not.

In SciPy, we can use the ttest_ind() function.

The test is described as:

> *Calculates the T-test for the means of two independent samples of scores.*

The null hypothesis of the test (H0) or the default expectation is that both samples were drawn from it means that there is no significant difference between the means.

If we get a p-value of <= 0.05, it means that we can reject the null hypothesis and that the means are at means for 95 similar samples out of 100, the means would be significantly different, and not so in 5 c

An important assumption of this statistical test, besides the data being Gaussian, is that both distributions have the same variance. We know this to be the case from reviewing the descriptive statistics in a previous step.

The complete code listing is provided below.

```
from pandas import read_csv
from scipy.stats import ttest_ind
from matplotlib import pyplot
```

[redacted]

```
# load results1
result1 = read_csv('results1.csv', header=None)
values1 = result1.values[:,0]
# load results2
result2 = read_csv('results2.csv', header=None)
values2 = result2.values[:,0]
# calculate the significance
value, pvalue = ttest_ind(values1, values2, equal_var=True)
print(value, pvalue)
if pvalue > 0.05:
    print('Samples are likely drawn from the same distributions (accept H0)')
else:
    print('Samples are likely drawn from different distributions (reject H0)')
```

Running the example prints the statistic and the p-value. We can see that the p-value is much lower than 0.05.

In fact, it is so small that we have a near certainty that the difference between the means is statistica

```
-22.7822655028 2.5159901708e-102
Samples are likely drawn from different distributions (reject H0)
```

Compare Means for Gaussian Results with Different Variar

What if the means were the same for the two sets of results, but the variance was different?

We would not be able to use the Student t-test as is. In fact, we would have to use a modified versio

In SciPy, this is the same ttest_ind() function, but we must set the "*equal_var*" argument to "*False*" to

We can demonstrate this with an example where we generate two sets of results with means that ar rd deviations (1 vs 10). We will generate 100 samples.

```
from numpy.random import seed
from numpy.random import normal
from scipy.stats import ttest_ind
# generate results
seed(1)
n = 100
values1 = normal(50, 1, n)
values2 = normal(51, 10, n)
# calculate the significance
```

```
value, pvalue = ttest_ind(values1, values2, equal_var=False)
print(value, pvalue)
if pvalue > 0.05:
    print('Samples are likely drawn from the same distributions (accept H0)')
else:
    print('Samples are likely drawn from different distributions (reject H0)')
```

Running the example prints the test statistic and the p-value.

We can see that there is good evidence (nearly 99%) that the samples were drawn from different distributions, that the means are significantly different.

```
-2.62233137406 0.0100871483783
Samples are likely drawn from different distributions (reject H0)
```

The closer the distributions are, the larger the sample that is required to tell them apart.

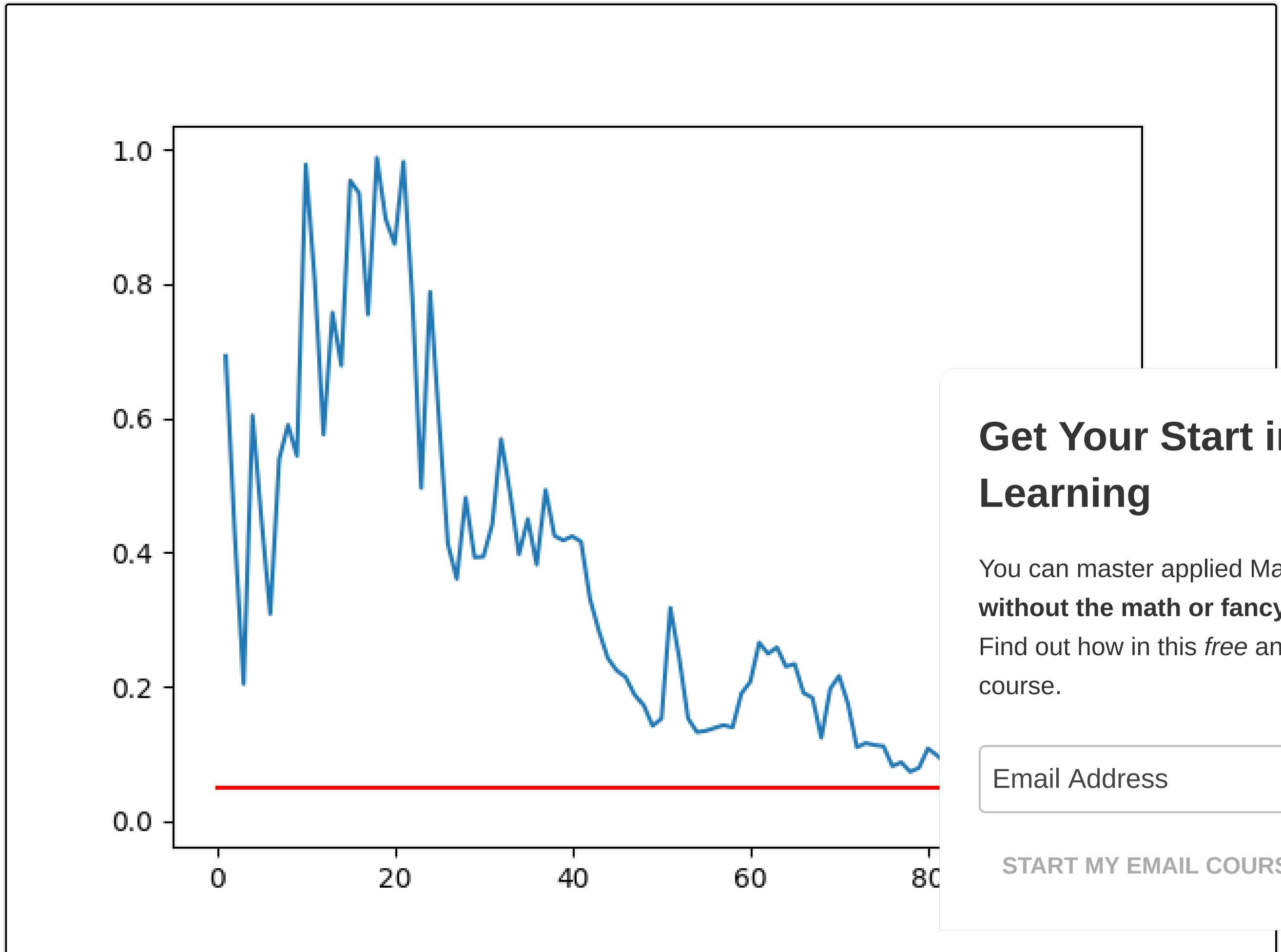
We can demonstrate this by calculating the statistical test on different sized sub-samples of each se
sample size.

We would expect the p-value to get smaller with the increase sample size. We can also draw a line a
sample size is large enough to indicate these two populations are significantly different.

```
from numpy.random import seed
from numpy.random import normal
from scipy.stats import ttest_ind
from matplotlib import pyplot
# generate results
seed(1)
n = 100
values1 = normal(50, 1, n)
values2 = normal(51, 10, n)
# calculate p-values for different subsets of results
pvalues = list()
for i in range(1, n+1):
    value, p = ttest_ind(values1[0:i], values2[0:i], equal_var=False)
    pvalues.append(p)
# plot p-values vs number of results in sample
pyplot.plot(pvalues)
# draw line at 95%, below which we reject H0
pyplot.plot([0.05 for x in range(len(pvalues))], color='red')
pyplot.show()
```

Running the example creates a line plot of p-value vs sample size.

We can see that for these two sets of results, the sample size must be about 90 before we have a 95% confidence that the means are significantly different (where the blue line intersects the red line).



Line Plot of p-value vs Sample Sizes

Compare Means for Non-Gaussian Results

We cannot use the Student t-test or the Welch's t-test if our data is not Gaussian.

An alternative statistical significance test we can use for non-Gaussian data is called the Kolmogoro

In SciPy, this is called the ks_2samp() function.

In the documentation, this test is described as:

> *This is a two-sided test for the null hypothesis that 2 independent samples are drawn from the same continuous distribution.*

This test can be used on Gaussian data, but will have less statistical power and may require large samples.

We can demonstrate the calculation of statistical significance on two sets of results with non-Gaussian distributions. We can generate two sets of results with overlapping uniform distributions (50 to 60 and 55 to 65). These sets of results will have different mean values of about 55 and 60 respectively.

The code below generates the two sets of 100 results and uses the Kolmogorov-Smirnov test to dem ion means is statistically significant.

```
from numpy.random import seed
from numpy.random import randint
from scipy.stats import ks_2samp
# generate results
seed(1)
n = 100
values1 = randint(50, 60, n)
values2 = randint(55, 65, n)
# calculate the significance
value, pvalue = ks_2samp(values1, values2)
print(value, pvalue)
if pvalue > 0.05:
    print('Samples are likely drawn from the same distributions (accept H0)')
else:
    print('Samples are likely drawn from different distributions (reject H0)')
```

Running the example prints the statistic and the p-value.

The p-value is very small, suggesting a near certainty that the difference between the two populations is significant.

```
0.47 2.16825856737e-10
Samples are likely drawn from different distributions (reject H0)
```

Further Reading

This section lists some articles and resources to dive deeper into the area of statistical significance [redacted]

- Normality test on Wikipedia
- Student's t-test on Wikipedia
- Welch's t-test on Wikipedia
- Kolmogorov–Smirnov test on Wikipedia

Summary

In this tutorial, you discovered how you can use statistical significance tests to interpret machine learning results.

You can use these tests to help you confidently choose one machine learning algorithm over another or one set of configuration parameters over another for the same algorithm.

You learned:

- How to use normality tests to check if your experimental results are Gaussian or not.
- How to use statistical tests to check if the difference between mean results is significant for Gau
- How to use statistical tests to check if the difference between mean results is significant for non-

Do you have any questions about this post or statistical significance tests?
Ask your questions in the comments below and I will do my best to answer.

Get Your Start in Machine Learning

About Jason Brownlee

Dr. Jason Brownlee is a husband, proud father, academic researcher, author, professional devel to helping developers get started and get good at applied machine learning. Learn more.

View all posts by Jason Brownlee →

< Estimate the Number of Experiment Repeats for Stochastic Machine Learning Algorithms

How to Use Weight Regularization with LSTM Networks for Time Series Forecasting >

10 Responses to *How to Use Statistical Significance Tests to Interpret Machine Learning Results*

David May 3, 2017 at 12:28 pm # REPLY

This is fantastic. Very clear and detailed explanation, and including the full code is much appreciated.

One question — do you know the corresponding functions in R? Specifically, what's the best way in R to test of a sample is normally distributed?

Thanks again for a great article

Jason Brownlee May 4, 2017 at 8:00 am #

Thanks David.

Yes, R can do this. I do not know the functions off-hand sorry.

Greg May 6, 2017 at 2:42 am #

You state when considering the boxplots that "The results for A look better than B" can you exp
plots are identical just offset by the difference in the mean.

Jason Brownlee May 6, 2017 at 7:48 am # REPLY

Exactly, the means are different and we are assuming an error score, so a lower mean value means a better result.

Andrew Kostandy May 6, 2017 at 8:21 pm # REPLY

Thanks Dr. Jason for the informative post. I have learned a lot from many of your posts!

I have a few comments and questions below.

1) I noticed that in both the sections “Compare Means for Gaussian Results” and “Compare Means for Gaussian Results with Different Variance” you have in the code: ‘pvalue = ttest_ind(values1, values2, equal_var=False’). Shouldn’t it be: ‘equal_var=True’ for the first section?

2) Also, in the section “Compare Means for Non-Gaussian Results”, you are using the Two-sample Kolmogorov-Smirnov test which according to Page 3 of the following document (http://www.mit.edu/~6.s085/notes/lecture5.pdf): “is sensitive to any differences at all in two distributions: two distributions with the same mean but significantly different shapes will produce a large value of Dn” [a significant result]. Therefore, getting a significant result only tells us they have different distributions but would not tell us which group is superior to the other as we’re not really comparing means as the section title indicates.

Instead of the Two-sample Kolmogorov-Smirnov test, wouldn’t it be better to do the Mann-Whitney U Tes
sample Kolmogorov-Smirnov test it is used to compare whether two groups have the same distribution. I
(http://www.mit.edu/~6.s085/notes/lecture5.pdf) states: “In contrast to the Kolmogorov-Smirnov test earli
(like its unpaired cousin the Mann-Whitney U) is only sensitive to changes in the median, and not to cha
‘conf.int=TRUE’ argument to the Mann-Whitney U Test (‘wilcox.test’ in R) which would allow you to deter
“median of pairwise differences” perspective (https://stats.stackexchange.com/questions/215764/signific
has-higher-median). There doesn’t seem to be a similar argument with the Two-sample Kolmogorov-Sm

3) When we are “comparing two different machine learning algorithms or when comparing the same algo
same group of 30 or 100 resamples for both algorithms or configurations, shouldn’t we use a paired t-tes
test (for non-gaussian results) instead? Effectively, we would be making 2 measurements (1 from each a
According to (http://emerald.tufts.edu/~gdallal/paired.htm): “When both measurements are made on the
eliminated from the comparison. The difference between treatments is compared to the way the differen
roughly the same for each subject, small treatment effects can be detected even if different subjects respond quite differently.” This also gives us the ability to
have a more powerful test using a smaller number of samples (algorithm runs on resamples) which saves computational time.

Get Your Start in Machine Learning

×

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Jason Brownlee May 7, 2017 at 5:38 am # REPLY

Hi Andrew,

Get Your Start in Machine Learning

[redacted]

Yes, that was a typo re the equal_var argument. Fixed.

Yes great suggestion. I am a big fan of reporting medians and using the Man-whitney U test.

Perhaps.

Viktor May 7, 2017 at 1:02 am # REPLY

Very informative, thank you!

Jason Brownlee May 7, 2017 at 5:42 am #

I'm glad you found it useful Viktor.

Roberto May 11, 2017 at 7:37 pm #

Hi Jason! Really useful article. Many thanks.

I want to ask: how exactly can you use this to select between models? I guess you use k-fold cross-valic
make the comparison, but usually you don't create too many folds (5 or 10, maybe). Are these tests still

Thanks again!

Jason Brownlee May 12, 2017 at 7:40 am # REPLY

Great question Roberto.

Regardless of your resampling method (train/test or CV), evaluate your model many times (30+). Repeat with another algorithm or configuration, then compare the two populations of results.

[redacted]

Leave a Reply

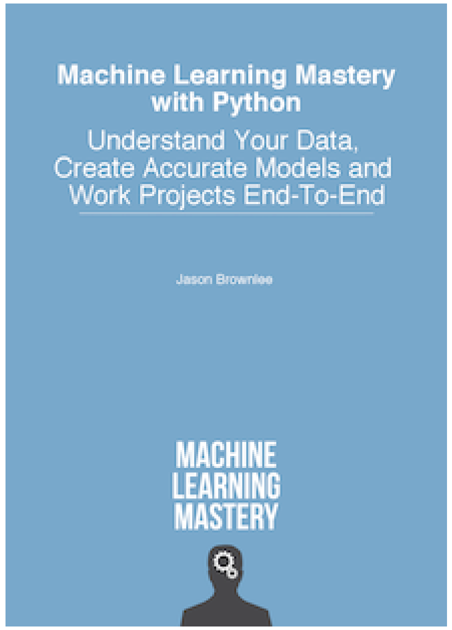
	Name (required)
	Email (will not be published) (required)
	Website

SUBMIT COMMENT

Welcome to Machine Learning Mastery

Hi, I'm Dr. Jason Brownlee.
My goal is to make practitioners like YOU awesome at applied machine learning.

Read More



POPULAR

Time Series Prediction with LSTM Recurrent Neural Networks in Python with Keras
JULY 21, 2016

Your First Machine Learning Project in Python Step-By-Step
JUNE 10, 2016

Develop Your First Neural Network in Python With Keras Step-By-Step
MAY 24, 2016

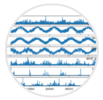
Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras
JULY 26, 2016

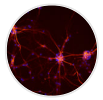
How to Setup a Python Environment for Machine Learning and Deep Learning with Anaconda
MARCH 13, 2017

Time Series Forecasting with the Long Short-Term Memory Network in Python
APRIL 7, 2017

Multi-Class Classification Tutorial with the Keras Deep Learning Library
JUNE 2, 2016

Regression Tutorial with the Keras Deep Learning Library in Python
JUNE 9, 2016


Multivariate Time Series Forecasting with LSTMs in Keras
AUGUST 14, 2017


How to Implement the Backpropagation Algorithm From Scratch In Python
NOVEMBER 7, 2016

Get Your Start in Machine Learning

×

You can master applied Machine Learning **without the math or fancy degree**.
Find out how in this *free* and *practical* email course.

Email Address

START MY EMAIL COURSE

Get Your Start in Machine Learning