learn-co-curriculum / dsc-in-depth-ab-testing-lab Public

View license

0 stars 204 forks

Star Watch

Code Issues Pull requests Actions Projects Security Insights

solution

This branch is 6 commits ahead, 8 commits behind master.

hoffm386 fix objectives spacing ... on Jul 27 8

View code

README.md

In Depth A/B Testing - Lab

Introduction

In this lab, you'll explore a survey from Kaggle regarding budding data scientists. With this, you'll form some initial hypotheses, and test them using the tools you've acquired to date.

Objectives

You will be able to:

- Conduct t-tests and an ANOVA on a real-world dataset and interpret the results

Load the Dataset and Perform a Brief Exploration

The data is stored in a file called **multipleChoiceResponses_cleaned.csv**. Feel free to check out the original dataset referenced at the bottom of this lab, although this cleaned version will undoubtedly be easier to work with. Additionally, meta-data regarding the questions is stored in a file name **schema.csv**. Load in the data itself as a Pandas DataFrame, and take a moment to briefly get acquainted with it.

> Note: If you can't get the file to load properly, try changing the encoding format as in `encoding='latin1'`

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style('darkgrid')
%matplotlib inline

import warnings
warnings.filterwarnings("ignore")
```

```
df = pd.read_csv('multipleChoiceResponses_cleaned.csv', encoding='latin1')
df.head()
```

<style scoped> .dataframe tbody tr th:only-of-type { vertical-align: middle; }

```
.dataframe tbody tr th {
    vertical-align: top;
}

.dataframe thead th {
    text-align: right;
}
```

</style>

	GenderSelect	Country	Age	EmploymentStatus	StudentStatus	Lear
0	Non-binary, genderqueer, or gender non-conforming	NaN	NaN	Employed full-time	NaN	NaN

	GenderSelect	Country	Age	EmploymentStatus	StudentStatus	Lear
1	Female	United States	30.0	Not employed, but looking for work	NaN	NaN
2	Male	Canada	28.0	Not employed, but looking for work	NaN	NaN
3	Male	United States	56.0	Independent contractor, freelancer, or self-em...	NaN	NaN
4	Male	Taiwan	38.0	Employed full-time	NaN	NaN

5 rows × 230 columns

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 26394 entries, 0 to 26393
Columns: 230 entries, GenderSelect to AdjustedCompensation
dtypes: float64(15), object(215)
memory usage: 46.3+ MB
```

Wages and Education

You've been asked to determine whether education is impactful to salary. Develop a hypothesis test to compare the salaries of those with Master's degrees to those with Bachelor's degrees. Are the two statistically different according to your results?

> Note: The relevant features are stored in the 'FormalEducation' and 'AdjustedCompensation' features.

You may import the functions stored in the `flatiron_stats.py` file to help perform your hypothesis tests. It contains the stats functions that you previously coded: `welch_t(a,b)`, `welch_df(a, b)`, and `p_value(a, b, two_sided=False)`.

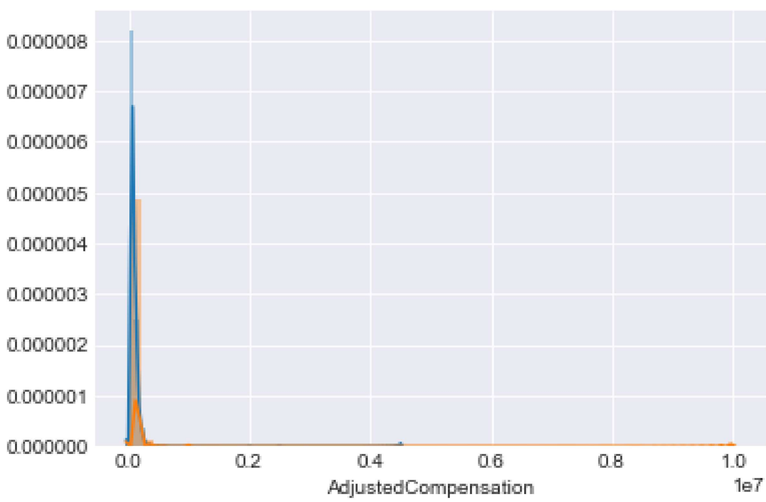
Note that `scipy.stats.ttest_ind(a, b, equal_var=False)` performs a two-sided Welch's t-test and that p-values derived from two-sided tests are two times the p-values derived from one-sided tests. See the documentation for more information.

```python
import flatiron_stats as fs
```

```python
#Subset the appropriate data into 2 groups
f1 = 'FormalEducation'
f2 = 'AdjustedCompensation'
f1c1 = "Master's degree"
f1c2 = "Bachelor's degree"
subset = df[(~df[f1].isnull()) & (~df[f2].isnull())]
s1 = subset[subset[f1]==f1c1][f2]
s2 = subset[subset[f1]==f1c2][f2]
```

```python
sns.distplot(s1)
sns.distplot(s2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x109bbd518>
```




```python
print('Comparison of {} for {} and {}'.format(f2, f1c1, f1c2))
print("Median Values: \ts1: {} \ts2: {}".format(round(s1.median(),2), round(s2.media
print("Mean Values: \ts1: {} \ts2: {}".format(round(s1.mean(),2), round(s2.mean(),2)
print('Sample sizes: \ts1: {} \ts2: {}'.format(len(s1), len(s2)))
print("Welch's t-test p-value:", fs.p_value_welch_ttest(s1, s2))
```

```
Comparison of AdjustedCompensation for Master's degree and Bachelor's degree
Median Values:  s1: 53812.17    s2: 38399.4
Mean Values:    s1: 69139.9     s2: 64887.1
```

```
Sample sizes:   s1: 1990        s2: 1107
Welch's t-test p-value: 0.33077639451272267
```

```
#Investigate Percentiles
for q in np.linspace(.8, 1, num=21):
    s1q =  round(s1.quantile(q=q), 2)
    s2q =  round(s2.quantile(q=q), 2)
    print('{}th percentile:\tset1: {}\tset2: {}'.format(round(q,2), s1q, s2q))
```

```
0.8th percentile:       set1: 103000.0  set2: 93233.13
0.81th percentile:      set1: 107009.0  set2: 95572.83
0.82th percentile:      set1: 110000.0  set2: 99276.38
0.83th percentile:      set1: 111503.83 set2: 100000.0
0.84th percentile:      set1: 115240.4  set2: 103040.0
0.85th percentile:      set1: 119582.6  set2: 105935.04
0.86th percentile:      set1: 120000.0  set2: 110000.0
0.87th percentile:      set1: 124719.88 set2: 112000.0
0.88th percentile:      set1: 129421.46 set2: 115000.0
0.89th percentile:      set1: 130000.0  set2: 120000.0
0.9th percentile:       set1: 135000.0  set2: 120346.5
0.91th percentile:      set1: 140000.0  set2: 126460.0
0.92th percentile:      set1: 149640.0  set2: 132615.4
0.93th percentile:      set1: 150000.0  set2: 140000.0
0.94th percentile:      set1: 160000.0  set2: 143408.8
0.95th percentile:      set1: 166778.6  set2: 150000.0
0.96th percentile:      set1: 180000.0  set2: 179849.74
0.97th percentile:      set1: 200000.0  set2: 195000.0
0.98th percentile:      set1: 211100.0  set2: 200000.0
0.99th percentile:      set1: 250000.0  set2: 250000.0
1.0th percentile:       set1: 4498900.0 set2: 9999999.0
```

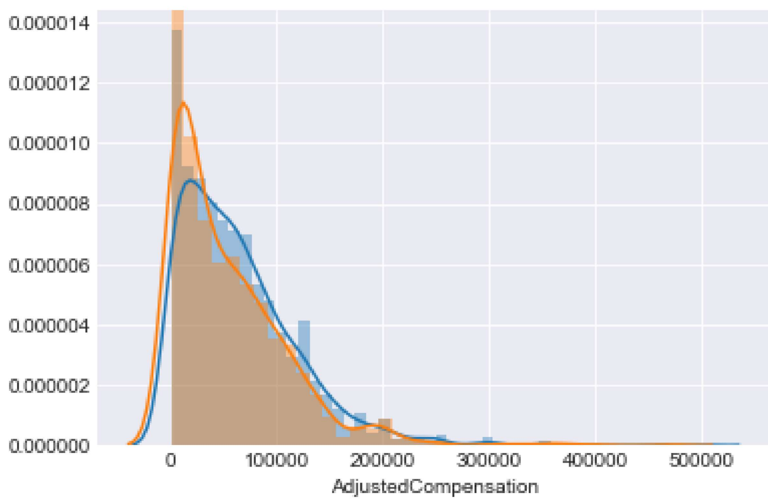
```
print('Repeated Test with Outliers Removed:')
print('S1: {}\tS2: {}'.format(f1c1, f1c2))
outlier_threshold = 500000
s1 = subset[(subset[f1]==f1c1) & (subset[f2]<=outlier_threshold)][f2]
s2 = subset[(subset[f1]==f1c2) & (subset[f2]<=outlier_threshold)][f2]
print("Median Values: \ts1: {} \ts2: {}".format(round(s1.median(),2), round(s2.media
print("Mean Values: \ts1: {} \ts2: {}".format(round(s1.mean(),2), round(s2.mean(),2)
print('Sample sizes: \ts1: {} \ts2: {}'.format(len(s1), len(s2)))
print("Welch's t-test p-value with outliers removed:", fs.p_value_welch_ttest(s1, s2
```

```
Repeated Test with Ouliers Removed:
S1: Master's degree     S2: Bachelor's degree
```

```
Median Values:  s1: 53539.72    s2: 38292.15
Mean Values:    s1: 63976.63    s2: 53744.35
Sample sizes:   s1: 1985        s2: 1103
Welch's t-test p-value with outliers removed: 4.4874583271514723e-07
```

```
sns.distplot(s1)
sns.distplot(s2)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1a10772a58>
```



Wages and Education II

Now perform a similar statistical test comparing the AdjustedCompensation of those with Bachelor's degrees and those with Doctorates. If you haven't already, be sure to explore the distribution of the AdjustedCompensation feature for any anomalies.

```
f1 = 'FormalEducation'
f2 = 'AdjustedCompensation'
subset = df[(~df[f1].isnull()) & (~df[f2].isnull())]
s1 = subset[subset[f1]=="Doctoral degree"][f2]
s2 = subset[subset[f1]=="Bachelor's degree"][f2]
print("Median Values: \ns1:{} \ns2:{}".format(round(s1.median(),2), round(s2.median(
print('Sample sizes: \ns1: {} \ns2: {}'.format(len(s1), len(s2)))
print("Welch's t-test p-value:", fs.p_value_welch_ttest(s1, s2))

print('\n\nRepeated Test with Ouliers Removed:')
outlier_threshold = 500000
s1 = subset[(subset[f1]=="Doctoral degree") & (subset[f2]<=outlier_threshold)][f2]
s2 = subset[(subset[f1]=="Bachelor's degree") & (subset[f2]<=outlier_threshold)][f2]
```

```
print('Sample sizes: \ns1: {} \ns2: {}'.format(len(s1), len(s2)))
print("Welch's t-test p-value with outliers removed:", fs.p_value_welch_ttest(s1, s2
```

```
Median Values:
s1:74131.92
s2:38399.4
Sample sizes:
s1: 967
s2: 1107
Welch's t-test p-value: 0.1568238199472023


Repeated Test with Ouliers Removed:
Sample sizes:
s1: 964
s2: 1103
Welch's t-test p-value with outliers removed: 0.0
```

Wages and Education III

Remember the multiple comparisons problem; rather than continuing on like this, perform an ANOVA test between the various 'FormalEducation' categories and their relation to 'AdjustedCompensation'.

```
#Perform ANOVA here
import statsmodels.api as sm
from statsmodels.formula.api import ols

formula = '{} ~ C({})'.format(f2, f1)
lm = ols(formula, df).fit()
table = sm.stats.anova_lm(lm, typ=2)
print(table)
```

```
                          sum_sq      df         F    PR(>F)
C(FormalEducation)  6.540294e+17     6.0  0.590714  0.738044
Residual            7.999414e+20  4335.0       NaN       NaN
```

```
temp = df[df[f2]<=5*10**5]
formula = '{} ~ C({})'.format(f2, f1)
lm = ols(formula, temp).fit()
```

```
table = sm.stats.anova_lm(lm, typ=2)
print(table)
```

```
                           sum_sq      df          F        PR(>F)
C(FormalEducation)   5.841881e+11     6.0  29.224224  1.727132e-34
Residual             1.439270e+13  4320.0        NaN           NaN
```

Additional Resources

Here's the original source where the data was taken from:
Kaggle Machine Learning & Data Science Survey 2017

Summary

In this lab, you practiced conducting actual hypothesis tests on actual data. From this, you saw how dependent results can be on the initial problem formulation, including preprocessing!

Releases

No releases published

Packages

No packages published

Contributors 4

mathymitchell

hoffm386 Erin R Hoffman

alexgriff Alex Griffith

lmcm18

Languages

- **Jupyter Notebook** 78.9%
- **Python** 21.1%