Week 6 lab

COGS 108, 9:00-9:50AM (B01)

https://github.com/alexavndra/cogs108-b01

Reminders!!

- A2 is due TODAY at 11:59PM!
- D5 is due MONDAY, NOVEMBER 13th at 11:59PM
- Office hours is still ongoing!
  - https://calendly.com/alexandrarh/office-hours
- Data Checkpoint due November 15th at 11:59PM
  - Understand the feedback received in the project proposal (earn back points)
  - Use TA/Professor OH to discuss on the feedback

Want to see a topic more/less covered? Let us know with this survey!

Sentiment analysis????

Sentiment analysis: what is it giving?

"The process of analyzing digital text to determine if the emotional tone of the message is positive, negative, or neutral" (aws.org)

- ➢ Denotation vs. connotation
- ➢ Context: Objectivity vs. subjectivity
  - Objective: My cat loves her food
  - Subjective: My cat loves her gravy slop
- ➢ Useful in recommending systems (e.g. Spotify DJ!)

Read more here:
en.wikipedia.com/wiki/Sentiment_analysis

D5: Inference

Part I : Data & Wrangling

df['height'] = df['height'].apply(function)

The `apply()` method allows you to apply a function along one of the axis of the DataFrame, default 0, which is the index (row) axis.

Syntax

```
dataframe.apply(func, axis, raw, result_type, args, kwds)
```

Parameters

The `axis`, `raw`, `result_type`, and `args` parameters are keyword arguments.

Parameter	Value	Description
func		Required. A function to apply to the DataFrame.

axis	`0`
	`1`
	`'index'`
	`'columns'`
		Optional, Which axis to apply the function to. default 0.

raw	`True` `False`	Optional, default False. Set to true if the row/column should be passed as an ndarray object

result_type	`'expand'`
	`'reduce'`
	`'broadcast'`
	`None`
		Optional, default None. Specifies how the result will be returned

args	*a tuple*	Optional, arguments to send into the function
kwds	*keyword arguments*	Optional, keyword arguments to send into the function

Part II : EDA

fig = pd.plotting.scatter_matrix(df[['column_1', column_2']])

A scatter matrix is useful because it allows you to visualize the relationship between multiple variables in a dataset at once.

pandas.plotting.scatter_matrix

```
pandas.plotting.scatter_matrix(frame, alpha=0.5, figsize=None, ax=None, grid=False, diagonal='hist', marker='.', density_kwds=None, hist_kwds=None, range_padding=0.05, **kwargs)
```
[source]

Draw a matrix of scatter plots.

Parameters:

frame : *DataFrame*

alpha : *float, optional*

Amount of transparency applied.

figsize : *(float,float), optional*

A tuple (width, height) in inches.

ax : *Matplotlib axis object, optional*

grid : *bool, optional*

Setting this to True will show the grid.

diagonal : *{'hist', 'kde'}*

Pick between 'kde' and 'hist' for either Kernel Density Estimation or Histogram plot in the diagonal.

marker : *str, optional*

Matplotlib marker type, default '.'.

density_kwds : *keywords*

Keyword arguments to be passed to kernel density estimate plot.

hist_kwds : *keywords*

Keyword arguments to be passed to hist function.

range_padding : *float, default 0.05*

Relative extension of axis range in x and y with respect to (x_max - x_min) or (y_max - y_min).

****kwargs**

Keyword arguments to be passed to scatter function.

Returns:

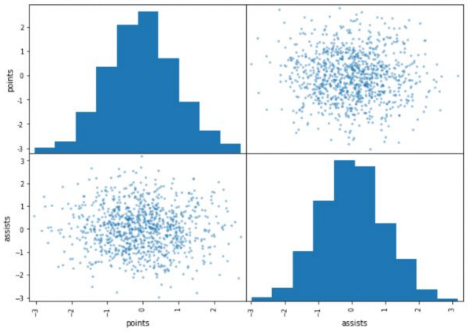
numpy.ndarray

A matrix of scatter plots.

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

#make this example reproducible
np.random.seed(0)

#create DataFrame
df = pd.DataFrame({'points': np.random.randn(1000),
                   'assists': np.random.randn(1000),
                   'rebounds': np.random.randn(1000)})

#view first five rows of DataFrame
df.head()
```

	points	assists	rebounds
0	1.764052	0.555963	-1.532921
1	0.400157	0.892474	-1.711970
2	0.978738	-0.422315	0.046135
3	2.240893	0.104714	-0.958374
4	1.867558	0.228053	-0.080812

```python
pd.plotting.scatter_matrix(df.iloc[:, 0:2])
```



Ref: https://pandas.pydata.org/docs/reference/api/pandas.plotting.scatter_matrix.html

Part III : ttest_ind

t_val, p_val = ttest_ind(df1, df_2)

ttest_ind used to check whether the unknown population means of given pair of groups are equal.

tt allows one to test the null hypothesis that the means of two groups are equal

scipy.stats.ttest_ind

```
scipy.stats.ttest_ind(a, b, axis=0, equal_var=True, nan_policy='propagate', permutations=None, random_state=None, alternative='two-sided', trim=0, *, keepdims=False)
```

[source]

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

This is a test for the null hypothesis that 2 independent samples have identical average (expected) values. This test assumes that the populations have identical variances by default.

Parameters: **a, b** : *array_like*

The arrays must have the same shape, except in the dimension corresponding to *axis* (the first, by default).

axis : *int or None, default: 0*

If an int, the axis of the input along which to compute the statistic. The statistic of each axis-slice (e.g. row) of the input will appear in a corresponding element of the output. If `None`, the input will be raveled before computing the statistic.

equal_var : *bool, optional*

If True (default), perform a standard independent 2 sample test that assumes equal population variances [1]. If False, perform Welch's t-test, which does not assume equal population variance [2].

New in version 0.11.0.

nan_policy : *{'propagate', 'omit', 'raise'}*

Defines how to handle input NaNs.

- `propagate`: if a NaN is present in the axis slice (e.g. row) along which the statistic is computed, the corresponding entry of the output will be NaN.
- `omit`: NaNs will be omitted when performing the calculation. If insufficient data remains in the axis slice along which the statistic is computed, the corresponding entry of the output will be NaN.
- `raise`: if a NaN is present, a `ValueError` will be raised.

Ref: https://www.geeksforgeeks.org/countplot-using-seaborn-in-python/

Part III : patsy.dmatrices

outcome_1, predictors_1 = patsy.dmatrices('y ~ x', df_sub)

Patsy uses R-style formulas to define the model and takes care of transforming the data for you.

We can pass the relation among variables as strings.
For Ex.
'y~x'
Or
'y~ x1 + x2 + b*x3'

4. Patsy

Patsy is a neat API to transform your data into experimentation model form. For regression and classification problems, you often want your data in the `Xy` form where `X` is a matrix (independent variable) and `y` is a column vector (dependent variable). In regression, let's say you have x_1, x_2 as your independent variable and y as your dependent variable. You might want to express the possible models as follows.

- $y = b_0 + x_1 + x_2$
- $y = b_0 + x_1 + x_2 + x_1 x_2$
- $y = b_0 + x_1 + x_2 + x_1^2 + x_2^2$
- $y = b_0 + x_1 + x_2 + x_1^2 + x_2^2 + x_1 x_2$

[3]:
```
from patsy import dmatrices

formula = 'y ~ height + weight + I(height**2) + I(weight**2) + height:weight'
y, X = dmatrices(formula, df, return_type='dataframe')

X
```

[3]:

	Intercept	height	weight	I(height ** 2)	I(weight ** 2)	height:weight
0	1.0	10.0	88.0	100.0	7744.0	880.0
1	1.0	20.0	99.0	400.0	9801.0	1980.0
2	1.0	30.0	125.0	900.0	15625.0	3750.0
3	1.0	40.0	155.0	1600.0	24025.0	6200.0
4	1.0	50.0	120.0	2500.0	14400.0	6000.0

Ref: https://www.geeksforgeeks.org/countplot-using-seaborn-in-python/

Part III : statsmodels.regression.linear_model.OLS

```
# Now use statsmodels to initialize an OLS linear model
# This step initializes the model, and provides the data (but does not actually compute the model)
mod_log = sm.OLS(outcome, predictors)

# fit the model
res_log = mod_log.fit()

# Check out the results
print(res_log.summary())
```

Introduction :

A linear regression model establishes the relation between a dependent variable(**y**) and at least one independent variable(**x**) as :

$\hat{y} = b_1 x + b_0$

In *OLS* method, we have to choose the values of $b_1$and $b_0$such that, the total sum of squares of the difference between the calculated and observed values of y, is minimised.

Formula for OLS:

$$S = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - b_1 x_1 - b_0)^2 = \sum_{i=1}^{n}(\hat{\epsilon}_i)^2 = min$$

Where,

$\hat{y}_i$= predicted value for the ith observation

y_i= actual value for the ith observation

ϵ_i= error/residual for the ith observation

n = total number of observations

To get the values of $b_0$and $b_1$which minimise S, we can take a partial derivative for each coefficient and equate it to zero.

Ref: https://www.geeksforgeeks.org/countplot-using-seaborn-in-python/

D5 Demo