Pyladies - 15.05.2017 - hands on data

May 15, 2017

1 dane z githuba

https://github.com/mu-opentrainings/PyLadies_data_manipulation
1.collaboration
podajcie mi swoj nick z githuba
git clone
przejdzcie do katalogu
stworzcie w katalogu swoj plik (koncowka _ksywka/inicjaly)
po zakonczeniu zajec
git add
git commit -m
git push
2.inna opcja fetch/pull request
https://gist.github.com/Chaser324/ce0505fbed06b947d962

2 pytanie an ktore chcemy odpowiedziec

Czy poów dzikich ryb korelowa z poziomem smierci samobojczych w latach 1950-2010/mial wplyw na poziom smierci samobojczych w latach 1950-2010?

3 dane

znalezienie dobrych danych to tez wyczyn
https://www.cdc.gov/nchs/hus/mentalhealth.htm
World Wild Fish Catch and Farmed Fish Production, 1950-2012
http://www.earth-policy.org/data_center/C26

```
In [13]: # sprawdzenie i sprzatanie danych
         # przypomnienie
         import pandas as pd
         df = pandas.read_excel(io, sheetname=0, header=0, skiprows=None,
                                skip_footer=0, index_col=None, names=None, parse_cols=None, parse_dat
                                date_parser=None, na_values=None, thousands=None, convert_float=True,
                                converters=None, dtype=None, true_values=None, false_values=None, eng
                                squeeze=False, **kwds)
         df.columns # nazwy kolumn
```

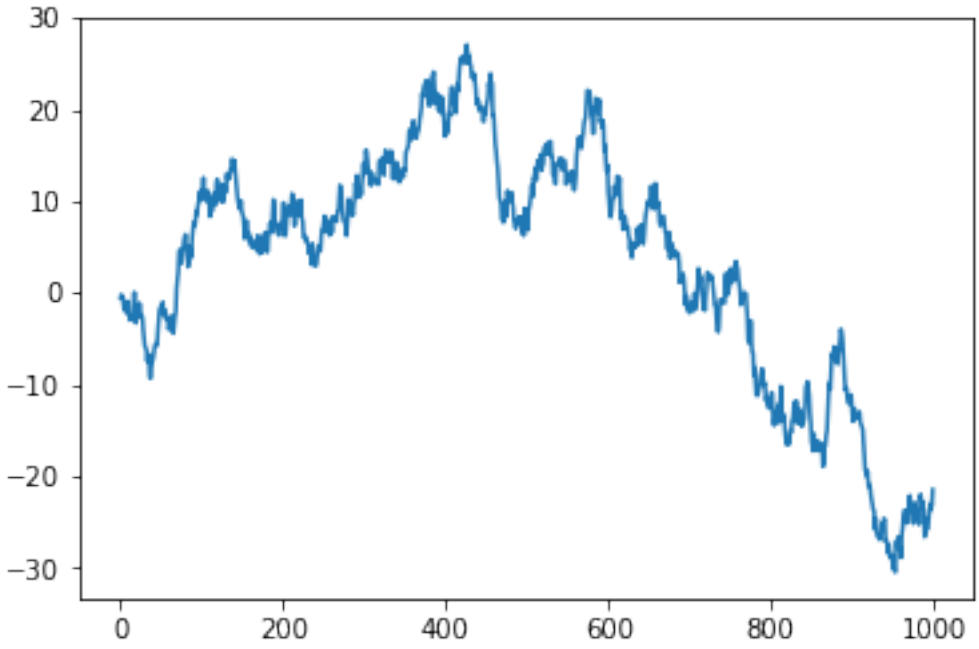
```
df.index.name # nazwa kolumny z indeksami
df.index # indeksy
df.dropna(axis=, thresh=, inplace=) # usuniecie wierszy z NaN
```

In [5]:
```
# uporzadkowanie danych, polaczenie tabeli w celu odpowiedzi na pytanie
w jednej tabeli lata w kolumnach, w drugiej w indeksie?

df.transpose() # zmienia kolumny z wierszami
df.loc[[rows], [columns]] # wyobor konkretnych danych z tabeli
pandas.concat(objs, axis=0, join='outer', join_axes=None, ignore_index=False,
              keys=None, levels=None, names=None, verify_integrity=False, copy=True)

# przy laczeniu danych pamietajcie o typie danych indeksu! sprawdzcie jakiego typu sa in
```

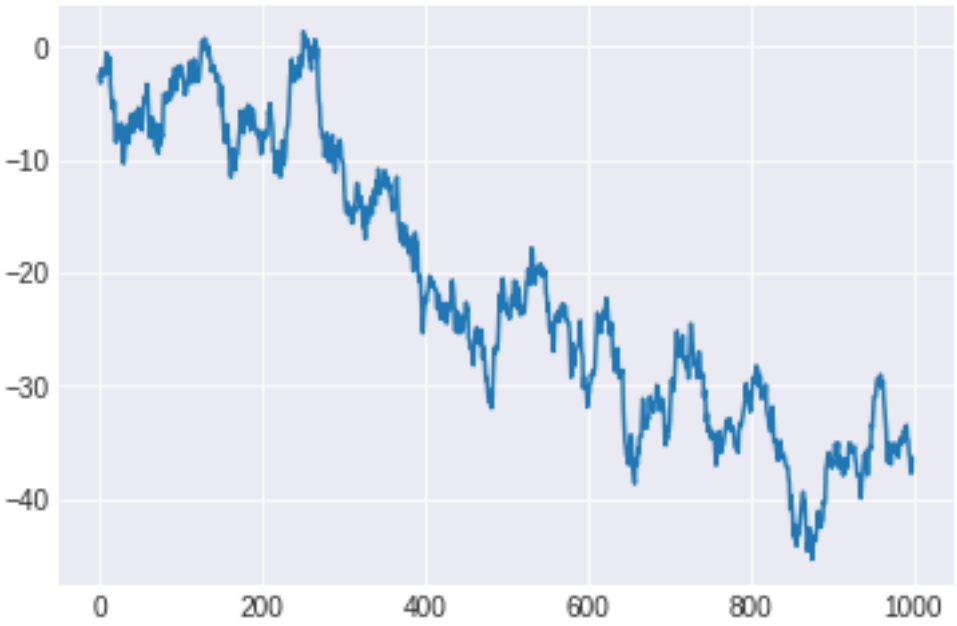
In [51]:
```
# wizualizacja danych
# liniowy wykres z tytulem, opisem osi x,y, legenda, kolor linii ryb niebieski, kolor l
# na jednym wykresie obie linie!
# matplotlib
import matplotlib.pyplot as plt
plt.plot(x, y, label=, c=)
plt.title()
plt.xlabel()
plt.ylabel()
plt.legend()
plt.show()
```

```
  File "<ipython-input-51-707f5e46eb4b>", line 6
plt.plot(x, y, label=, c=)
                      ^
SyntaxError: invalid syntax
```

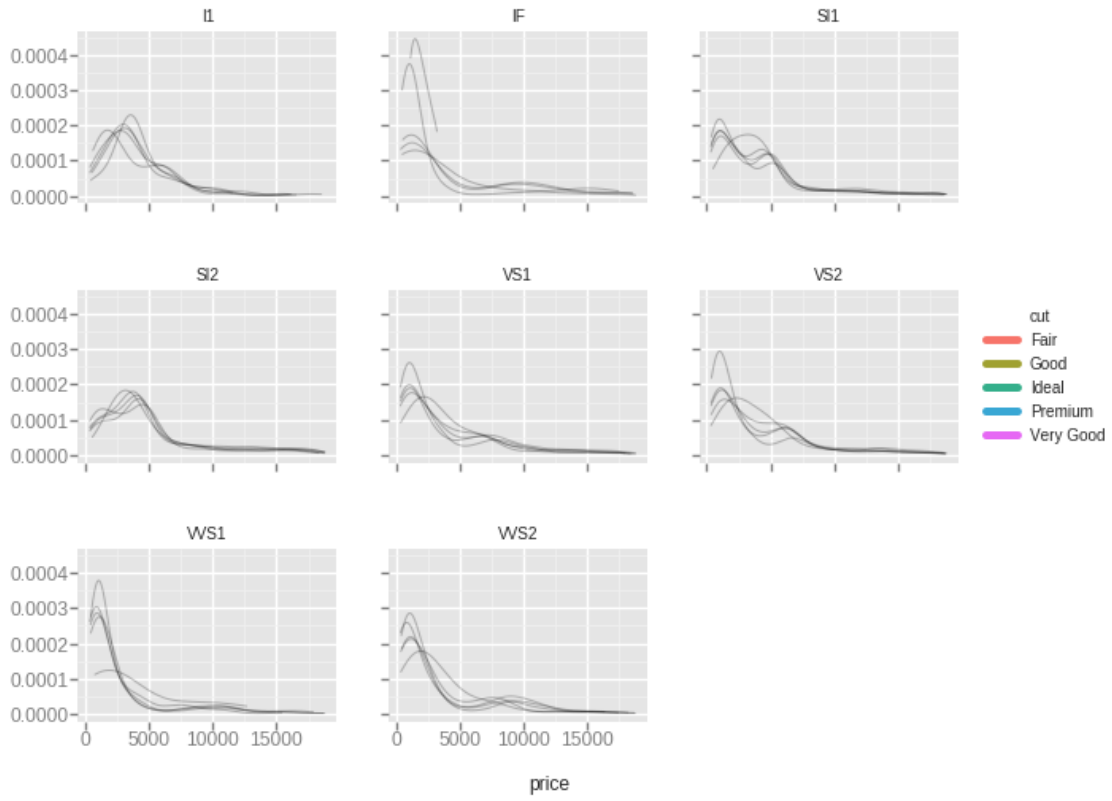
In [61]:
```
# inne biblioteki do wizualizacji danych
# seaborn
# seaborn
import seaborn as sns
# pracuje na matplotlibie
sns.reset_orig()
plt.plot(np.cumsum(np.random.randn(1000,1)))
plt.show()
```



In [63]:
```
sns.set_style("darkgrid")
plt.plot(np.cumsum(np.random.randn(1000,1)))
plt.show()
```



In [66]:
```
# dla tych co lubia R
# ggplot based on R's ggplot2
from ggplot import *

ggplot(diamonds, aes(x='price', fill='cut')) +\
    geom_density(alpha=0.25) +\
    facet_wrap("clarity")
```



Out[66]: <ggplot: (8729521379855)>

In [1]:
```
# analiza statystyczna
# corelacja Pearsona
from scipy.stats.stats import pearsonr
help(pearsonr)

# dla zainteresowanych statystyka https://docs.scipy.org/doc/scipy/reference/stats.html
```

```
Help on function pearsonr in module scipy.stats.stats:

pearsonr(x, y)
    Calculates a Pearson correlation coefficient and the p-value for testing
    non-correlation.
```

```
    The Pearson correlation coefficient measures the linear relationship
    between two datasets. Strictly speaking, Pearson's correlation requires
    that each dataset be normally distributed, and not necessarily zero-mean.
    Like other correlation coefficients, this one varies between -1 and +1
    with 0 implying no correlation. Correlations of -1 or +1 imply an exact
    linear relationship. Positive correlations imply that as x increases, so
    does y. Negative correlations imply that as x increases, y decreases.

    The p-value roughly indicates the probability of an uncorrelated system
    producing datasets that have a Pearson correlation at least as extreme
    as the one computed from these datasets. The p-values are not entirely
    reliable but are probably reasonable for datasets larger than 500 or so.

    Parameters
    ----------
    x : (N,) array_like
        Input
    y : (N,) array_like
        Input

    Returns
    -------
    r : float
        Pearson's correlation coefficient
    p-value : float
        2-tailed p-value

    References
    ----------
    http://www.statsoft.com/textbook/glosp.html#Pearson%20Correlation
```

```
In [ ]: # czy dane maja rozklad normalny?
        scipy.stats.mstats.normaltest(data)
```

```
In [52]: # zadanie: a teraz sprawdcie (a wykresie i korelacj Pearsona)
         # czy istnieje zaleno midzy poowem dzikich ryb a samobójstwami mlodziezy w wieku 15-24
```

4 podsumowanie

pamietajcie o wyslaniu pliku na gita

5 Python in machine learning

kopalnia wiedzy o machine learning:

http://www.kdnuggets.com/
kursy:
Coursera - Applied Machine Learning in Python
DataCamp - e.g. Supervised Learning with scikit-learn
Udemy - Machine Learning A-Z: Hands-On Python & R In Data Science
podcasts:
https://dataskeptic.com/podcast

machine learning - uczenie machynowe -> moliwo uczenia si komputera na podstawie danych bez specyficznego 'zakodowania' algorytmu supervised vs unsupervised machine learning supervised (nadzorowane uczenie) - wyjcia s okrelone, np. maile typu 'spam' i 'niespam' unsupervised (nienadzorowane uczenie) - nie wiemy czego spodziewa si na kocu, nie ma okrelonego wyjcia, np. wykrywanie nieokrelonych wzorów (patterns) w danych; np. klastrowanie i wiele wiele innych

jedna z popularnych bibliotek do uczenia nadzorowanego scikit-learn/sklearn (inne: tensorflow, keras)

dane wejciowe: zmienne predykcyjne dane wyjciowe: zmienna docelowa (klasyfikacja - okrelone grupy lub regresja - zmienna liczbowa)

```
In [15]: #przyklad klasyfikacji

         from sklearn import datasets
         import pandas as pd
         import numpy as np
         import matplotlib.pyplot as plt

In [17]: iris = datasets.load_iris()
         print(iris)
```

```
{'feature_names': ['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (c
       [ 4.9,  3. ,  1.4,  0.2],
       [ 4.7,  3.2,  1.3,  0.2],
       [ 4.6,  3.1,  1.5,  0.2],
       [ 5. ,  3.6,  1.4,  0.2],
       [ 5.4,  3.9,  1.7,  0.4],
       [ 4.6,  3.4,  1.4,  0.3],
       [ 5. ,  3.4,  1.5,  0.2],
       [ 4.4,  2.9,  1.4,  0.2],
       [ 4.9,  3.1,  1.5,  0.1],
       [ 5.4,  3.7,  1.5,  0.2],
       [ 4.8,  3.4,  1.6,  0.2],
       [ 4.8,  3. ,  1.4,  0.1],
       [ 4.3,  3. ,  1.1,  0.1],
       [ 5.8,  4. ,  1.2,  0.2],
       [ 5.7,  4.4,  1.5,  0.4],
       [ 5.4,  3.9,  1.3,  0.4],
       [ 5.1,  3.5,  1.4,  0.3],
       [ 5.7,  3.8,  1.7,  0.3],
       [ 5.1,  3.8,  1.5,  0.3],
       [ 5.4,  3.4,  1.7,  0.2],
```

```
[ 5.1,  3.7,  1.5,  0.4],
[ 4.6,  3.6,  1. ,  0.2],
[ 5.1,  3.3,  1.7,  0.5],
[ 4.8,  3.4,  1.9,  0.2],
[ 5. ,  3. ,  1.6,  0.2],
[ 5. ,  3.4,  1.6,  0.4],
[ 5.2,  3.5,  1.5,  0.2],
[ 5.2,  3.4,  1.4,  0.2],
[ 4.7,  3.2,  1.6,  0.2],
[ 4.8,  3.1,  1.6,  0.2],
[ 5.4,  3.4,  1.5,  0.4],
[ 5.2,  4.1,  1.5,  0.1],
[ 5.5,  4.2,  1.4,  0.2],
[ 4.9,  3.1,  1.5,  0.1],
[ 5. ,  3.2,  1.2,  0.2],
[ 5.5,  3.5,  1.3,  0.2],
[ 4.9,  3.1,  1.5,  0.1],
[ 4.4,  3. ,  1.3,  0.2],
[ 5.1,  3.4,  1.5,  0.2],
[ 5. ,  3.5,  1.3,  0.3],
[ 4.5,  2.3,  1.3,  0.3],
[ 4.4,  3.2,  1.3,  0.2],
[ 5. ,  3.5,  1.6,  0.6],
[ 5.1,  3.8,  1.9,  0.4],
[ 4.8,  3. ,  1.4,  0.3],
[ 5.1,  3.8,  1.6,  0.2],
[ 4.6,  3.2,  1.4,  0.2],
[ 5.3,  3.7,  1.5,  0.2],
[ 5. ,  3.3,  1.4,  0.2],
[ 7. ,  3.2,  4.7,  1.4],
[ 6.4,  3.2,  4.5,  1.5],
[ 6.9,  3.1,  4.9,  1.5],
[ 5.5,  2.3,  4. ,  1.3],
[ 6.5,  2.8,  4.6,  1.5],
[ 5.7,  2.8,  4.5,  1.3],
[ 6.3,  3.3,  4.7,  1.6],
[ 4.9,  2.4,  3.3,  1. ],
[ 6.6,  2.9,  4.6,  1.3],
[ 5.2,  2.7,  3.9,  1.4],
[ 5. ,  2. ,  3.5,  1. ],
[ 5.9,  3. ,  4.2,  1.5],
[ 6. ,  2.2,  4. ,  1. ],
[ 6.1,  2.9,  4.7,  1.4],
[ 5.6,  2.9,  3.6,  1.3],
[ 6.7,  3.1,  4.4,  1.4],
[ 5.6,  3. ,  4.5,  1.5],
[ 5.8,  2.7,  4.1,  1. ],
[ 6.2,  2.2,  4.5,  1.5],
```

```
[ 5.6,  2.5,  3.9,  1.1],
[ 5.9,  3.2,  4.8,  1.8],
[ 6.1,  2.8,  4. ,  1.3],
[ 6.3,  2.5,  4.9,  1.5],
[ 6.1,  2.8,  4.7,  1.2],
[ 6.4,  2.9,  4.3,  1.3],
[ 6.6,  3. ,  4.4,  1.4],
[ 6.8,  2.8,  4.8,  1.4],
[ 6.7,  3. ,  5. ,  1.7],
[ 6. ,  2.9,  4.5,  1.5],
[ 5.7,  2.6,  3.5,  1. ],
[ 5.5,  2.4,  3.8,  1.1],
[ 5.5,  2.4,  3.7,  1. ],
[ 5.8,  2.7,  3.9,  1.2],
[ 6. ,  2.7,  5.1,  1.6],
[ 5.4,  3. ,  4.5,  1.5],
[ 6. ,  3.4,  4.5,  1.6],
[ 6.7,  3.1,  4.7,  1.5],
[ 6.3,  2.3,  4.4,  1.3],
[ 5.6,  3. ,  4.1,  1.3],
[ 5.5,  2.5,  4. ,  1.3],
[ 5.5,  2.6,  4.4,  1.2],
[ 6.1,  3. ,  4.6,  1.4],
[ 5.8,  2.6,  4. ,  1.2],
[ 5. ,  2.3,  3.3,  1. ],
[ 5.6,  2.7,  4.2,  1.3],
[ 5.7,  3. ,  4.2,  1.2],
[ 5.7,  2.9,  4.2,  1.3],
[ 6.2,  2.9,  4.3,  1.3],
[ 5.1,  2.5,  3. ,  1.1],
[ 5.7,  2.8,  4.1,  1.3],
[ 6.3,  3.3,  6. ,  2.5],
[ 5.8,  2.7,  5.1,  1.9],
[ 7.1,  3. ,  5.9,  2.1],
[ 6.3,  2.9,  5.6,  1.8],
[ 6.5,  3. ,  5.8,  2.2],
[ 7.6,  3. ,  6.6,  2.1],
[ 4.9,  2.5,  4.5,  1.7],
[ 7.3,  2.9,  6.3,  1.8],
[ 6.7,  2.5,  5.8,  1.8],
[ 7.2,  3.6,  6.1,  2.5],
[ 6.5,  3.2,  5.1,  2. ],
[ 6.4,  2.7,  5.3,  1.9],
[ 6.8,  3. ,  5.5,  2.1],
[ 5.7,  2.5,  5. ,  2. ],
[ 5.8,  2.8,  5.1,  2.4],
[ 6.4,  3.2,  5.3,  2.3],
[ 6.5,  3. ,  5.5,  1.8],
```

```
        [ 7.7,  3.8,  6.7,  2.2],
        [ 7.7,  2.6,  6.9,  2.3],
        [ 6. ,  2.2,  5. ,  1.5],
        [ 6.9,  3.2,  5.7,  2.3],
        [ 5.6,  2.8,  4.9,  2. ],
        [ 7.7,  2.8,  6.7,  2. ],
        [ 6.3,  2.7,  4.9,  1.8],
        [ 6.7,  3.3,  5.7,  2.1],
        [ 7.2,  3.2,  6. ,  1.8],
        [ 6.2,  2.8,  4.8,  1.8],
        [ 6.1,  3. ,  4.9,  1.8],
        [ 6.4,  2.8,  5.6,  2.1],
        [ 7.2,  3. ,  5.8,  1.6],
        [ 7.4,  2.8,  6.1,  1.9],
        [ 7.9,  3.8,  6.4,  2. ],
        [ 6.4,  2.8,  5.6,  2.2],
        [ 6.3,  2.8,  5.1,  1.5],
        [ 6.1,  2.6,  5.6,  1.4],
        [ 7.7,  3. ,  6.1,  2.3],
        [ 6.3,  3.4,  5.6,  2.4],
        [ 6.4,  3.1,  5.5,  1.8],
        [ 6. ,  3. ,  4.8,  1.8],
        [ 6.9,  3.1,  5.4,  2.1],
        [ 6.7,  3.1,  5.6,  2.4],
        [ 6.9,  3.1,  5.1,  2.3],
        [ 5.8,  2.7,  5.1,  1.9],
        [ 6.8,  3.2,  5.9,  2.3],
        [ 6.7,  3.3,  5.7,  2.5],
        [ 6.7,  3. ,  5.2,  2.3],
        [ 6.3,  2.5,  5. ,  1.9],
        [ 6.5,  3. ,  5.2,  2. ],
        [ 6.2,  3.4,  5.4,  2.3],
        [ 5.9,  3. ,  5.1,  1.8]]), 'target_names': array(['setosa', 'versicolor', 'virginica'],
       dtype='<U10'), 'target': array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2,
       2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2, 2])}
```

```
In [18]: print(iris.keys())
```

```
dict_keys(['feature_names', 'DESCR', 'data', 'target_names', 'target'])
```

```
In [21]: print(iris.data.shape)
```

```
(150, 4)
```

```
In [22]: print(iris.target_names)
```

```
['setosa' 'versicolor' 'virginica']
```

```
In [23]: df = pd.DataFrame(iris.data, columns = iris.feature_names)
         df.head()
```

Out[23]:

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

```
In [27]: from sklearn.neighbors import KNeighborsClassifier
```

```
In [33]: knn = KNeighborsClassifier(n_neighbors=5)
         knn.fit(iris['data'], iris['target'])
```

```
Out[33]: KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                    metric_params=None, n_jobs=1, n_neighbors=5, p=2,
                    weights='uniform')
```

```
In [45]: iris['data'][0:150:40]
```

```
Out[45]: array([[ 5.1,  3.5,  1.4,  0.2],
                [ 5. ,  3.5,  1.3,  0.3],
                [ 5.5,  2.4,  3.8,  1.1],
                [ 6.9,  3.2,  5.7,  2.3]])
```

```
In [46]: iris['target'][0:150:40]
```

```
Out[46]: array([0, 0, 1, 2])
```

```
In [47]: knn.predict(iris['data'][0:150:40])
```

```
Out[47]: array([0, 0, 1, 2])
```

```
In [48]: knn.predict([[0, 4.0, 5.0, 6.0]])
```

```
Out[48]: array([2])
```

```
In [ ]:
```