Welcome to The Carpentries Etherpad!

This pad is synchronized as you type, so that everyone viewing this page sees the same text. This allows you to collaborate seamlessly on documents.

Use of this service is restricted to members of The Carpentries community; this is not for general purpose use (for that, try https://etherpad.wikimedia.org).

Users are expected to follow our code of conduct: https://docs.carpentries.org/topic_folders/policies/code-of-conduct.html



---

# Welcome to the ML with sklearn workshop!

## Important Links

Workshop Website : https://uw-madison-datascience.github.io/2021-09-15-uwmadison-mini/
Lesson : https://carpentries-incubator.github.io/machine-learning-novice-sklearn/
Setup : https://carpentries-incubator.github.io/machine-learning-novice-sklearn/setup.html
Feedback Form: https://forms.gle/wSRvADnx96NGwVWk6
Slides:
https://docs.google.com/presentation/d/1x8XajtMleQWXOksdnSQKDggYSJRQBVJX_eSUBm9jV_k/edit?usp=sharing

## Day 2

Sign-in
*Name, Program/Department/Organization*

- Sarah Stevens (she/her/hers), Data Science Hub
- Scott Wildman, Academic Planning and Institutional Research, HELPER
- Chris Kirby (he/him) , GLAS Education
- Trisha Adamus (she/her), HELPER
- Casey Schacher (she/her), Science & Engineering Libraries, HELPER
- Meredith Shapiro - American Family Insurance
- Bethany Moore (she/her), Botany Department, post-doc scientist
- Liliana Fadul - Animal and Dairy sciences department
- Lauren Baker, Scientist @ School of Veterinary Medicine, UW Madison
- Russell DImond, Social Science Computing Cooperative
- Siddharth Uppal - Pharmaceutical Sciences - Discover natural products from bacteria that can't be cultured in the lab

- Erwin Lares - Spanish & Portuguese
- Jean-Yves Sgro - French & some Spanish. (Sorry to be late, I had locked myself out of the office!)
- Steve Goldstein - (he / him) Data Science Institute and Botany Department, HELPER

**Notes:**

# Clustering Continued

## Using Silhouette Score To Choose Cluster Number

**# Copy the below code (lines 47 - 168) into jupyter notebook**

```
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm

# Generating the sample data from make_blobs
# This particular setting has one distinct cluster and 3 clusters placed close
# together.
X, y = make_blobs(
    n_samples=500,
    n_features=2,
    centers=4,
    cluster_std=.5,
    random_state=1, # For reproducibility
)

range_n_clusters = [2,3,4,5]

for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # Initialize the clusterer with n_clusters value and a random generator
```

```
# seed of 1 for reproducibility.
clusterer = KMeans(n_clusters=n_clusters, random_state=1)
cluster_labels = clusterer.fit_predict(X)

# The silhouette_score gives the average value for all the samples.
# This gives a perspective into the density and separation of the formed
# clusters
silhouette_avg = silhouette_score(X, cluster_labels)
print(
    "For n_clusters =",
    n_clusters,
    "The average silhouette_score is :",
    silhouette_avg,
)

# Compute the silhouette scores for each sample
sample_silhouette_values = silhouette_samples(X, cluster_labels)
y_lower = 10
for i in range(n_clusters):
    # Aggregate the silhouette scores for samples belonging to
    # cluster i, and sort them
    ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]

    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.nipy_spectral(float(i) / n_clusters)
    ax1.fill_betweenx(
        np.arange(y_lower, y_upper),
        0,
        ith_cluster_silhouette_values,
        facecolor=color,
        edgecolor=color,
        alpha=0.7,
    )

    # Label the silhouette plots with their cluster numbers at the middle
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Compute the new y_lower for next plot
    y_lower = y_upper + 10  # 10 for the 0 samples

ax1.set_title("The silhouette plot for the various clusters.")
ax1.set_xlabel("The silhouette coefficient values")
ax1.set_ylabel("Cluster label")

# The vertical line for average silhouette score of all the values
```

```
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

    # 2nd Plot showing the actual clusters formed
    colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
    ax2.scatter(
        X[:, 0], X[:, 1], marker=".", s=30, lw=0, alpha=0.7, c=colors, edgecolor="k"
    )

    # Labeling the clusters
    centers = clusterer.cluster_centers_
    # Draw white circles at cluster centers
    ax2.scatter(
        centers[:, 0],
        centers[:, 1],
        marker="o",
        c="white",
        alpha=1,
        s=200,
        edgecolor="k",
    )

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker="$%d$" % i, alpha=1, s=50, edgecolor="k")

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(
        "Silhouette analysis for KMeans clustering on sample data with n_clusters = %d"
        % n_clusters,
        fontsize=14,
        fontweight="bold",
    )

plt.show()
```

## Challenge

```
# What are the best and second best silhouette scores and their
# associated cluster numbers...
#  when cluster_std=0.5?
#  when cluster_std=1?
#  when cluster_std=2?
```

Russell
0.5: .819 (4), .749 (2)
1: .705 (2), .651 (4)
2: .602 (2), .443 (4)

Beth:
0.5: 4: 0.82, 2: 0.75
1: 2: 0.7, 4: 0.65
2: 2: 0.6, 4: 0.44

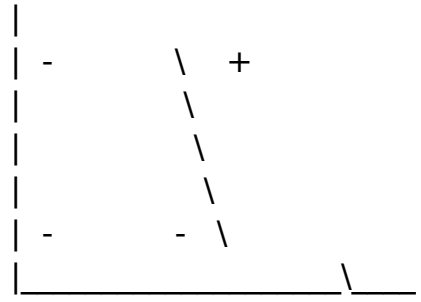
Meredith
STD = 0.5 -> 4 clusters (0.82), 2 clusters (0.75)
STD = 1-> 2 clusters (0.7), 4 clusters (0.65)
STD = 2 -> 2 clusters (0.6), 4 clusters (0.44)

Erwin
for sd = 0.5, 1st best is n = 4 (0.82), 2nd best n = 2 (0.75)
for sd = 1, 1st best is n = 2 (0.71), 2nd best n = 4 (0.65)
for sd = 2, 1st best is n = 2 (.060), 2nd best n = 4 (0.44)

Sidd
When sd=0.5, best is n=4 (0.818), 2nd best is n=2 (0.699)
When sd=1, best is n=2 (0.704), 2nd best is n=4 (0.650)
When sd=2, best is n=2 (0.601), 2nd best is n=4 (0.442)

# Are the differences in silhouette scores between the first and second best number of clusters large?
# How confident would you be in selecting cluster number at each standard deviation setting?

Here is an ASCII Version... (best with "code" font...)



Neural Networks

Open new python 3 jupyter notebook

# Coding Perceptron

import numpy as np

```
def perceptron(inputs, weights, threshold):
    assert len(inputs) == len(weights)

    # multiply the inputs and weights
    values = np.multiply(inputs, weights)

    # sum the results
    total = sum(values)

    # decide if value if we should activate perceptron
    if total < threshold:
        return 0
    else:
        return 1

# Set weights of perceptron to learn OR, AND, and NOT functions

# OR
inputs = [[0.0, 0.0], [1.0, 0.0], [0.0, 1.0], [1.0, 1.0]]
for input in inputs:
    weights = [1.0, 1.0]
    thresholds = 0.5
    print(input, perceptron(input, weights, threshold ))

# AND
inputs = [[0.0, 0.0], [1.0, 0.0], [0.0, 1.0], [1.0, 1.0]]
for input in inputs:
    weights = [ 0.5 , 0.5 ]
    thresholds = 1.0
    print(input, perceptron(input, weights, threshold ))

# NOT : We need a "bias term" which is always the same value
inputs = [[0.0, 1.0], [1.0, 1.0]]
for input in inputs:
    weights = [ -1.0 , 1.0  ]
    thresholds = 1.0
    print(input, perceptron(input, weights, threshold ))



# AND
inputs = [[0.0,0.0],[1.0,0.0],[0.0,1.0],[1.0,1.0]]
for input in inputs:
    weights=[.5,.5]
    threshold=1.0
    print(input,perceptron(input,weights,threshold))

# multilayer perceptrons in sklearn
import matplotlib.pyplot as plt
```

```
import sklearn.datasets as skl_data
import sklearn.neural_network as skl_nn

data, labels = skl_data.fetch_openml('mnist_784', version = 1, return_X_y = True)
data = data / 255.0

print(data.shape)

mlp = skl_nn.MLPClassifier(hidden_layer_sizes = (50, ), max_iter = 50, verbose = 1, random_state = 1)

data_train = data[0:60000]
labels_train = labels[0:60000]

data_test = data[60001:]
labels_test = labels[60001:]

# fit MLP model to train set
mlp.fit(data_train, labels_train)

print("Training set score", mlp.score(data_train, labels_train))
print("Testing set score", mlp.score(data_test, labels_test))

# openCV

import cv2
import matplotlib.pyplot as pyplot

#https://jspaint.app/
# image -> attributes -> Set width and height to 28, pixels, black and white (maybe missed some settings here)
# draw character -> Save it as digit.png -> move it to the intro_ML/data folder (probably on your desktop)

digit = cv2.imread("data/digit.png")
digit_gray = cv2.cv5Color(digit, cv2.COLOR_BGG2GRAY)
digit_norm = digit_gray / 255.0 # pixels range from 0-255, so normalizing them to be between 0-1
plt.matshow(digit_norm)
plt.show()
print("Your digit is", mlp.predict([digit_norm.reshape(784)]))
```

NOTE ADDED ON 12NOV2021: I had to change the line for digit_gray to :

**digit_gray = cv2.cvtColor(digit, cv2.COLOR_BGR2GRAY)**

i.e. 2 errors:
* change cv2.cv5Color to cv2.cvtColor (with a t instead of the 5)
* change cv2.COLOR_BGG2GRAY to cv2.COLOR_BGR2GRAY

To list all options use code:

```
import cv2
print(dir(cv2))
```

## Cross Validation

```
import matplotlib.pyplot as plt
import sklearn.datasets as skl_data
import sklearn.neural_network as skl_nn
import sklearn.model_selection as skl_msel

data, labels =  skl_data.fetch_openml('mnist_784', version = 1, return_X_y = True)
data = data / 255.0

# init MLP
mlp = skl_nn.MLPClassifier(hidden_layer_sizes = (50, ), max_iter = 50, verbose = 1, random_state = 1)

# initialize kfold split
kfold = skl_msel.KFold(n_splits = 5)

for (train_indices, test_indices) in kfold.split(data):
    data_train = data[train_indices]
    labels_train = labels[train_indices]

    data_test = data.iloc[test_indices, : ]
    labels_test = labels[test_indices]

    mlp.fit(data_train, labels_train)

    print("Training set score", mlp.score(data_train, labels_train))
    print("Testing set score", mlp.score(data_test, labels_test))
```

## Ethics of ML in Research

- Exercise: Think of a use case for machine learning in your research areas. What ethical implications (if any) might there be from using machine learning in your research? Write down your answers in the etherpad.

Lauren: I am not there yet, but I think it is possible that machine learning algorithms applied to my patient data (peoples' dogs--pets) could suggest one treatment modality over another, and that may or may not be whats best for the patient depending on how good the model is!

Russell: what I'm mostly seeing is machine learning as a form of model selection. Given that no one cares what social scientists think there's not much risk of affecting peoples' lives. :)

More seriously, the main risk is that data collection methods often systematically do a worse job with underprivileged people. Insofar as politicians are most likely to base policy decisions on social science

when it comes to "welfare" (defined broadly) that's a problem.

JYS: ML does not affect me or my job at the moment. But, while the ML biases might happen due to data collection, and the inherent ML methods, one might wonder if ML might be *purposely* used to fabricate situations, or manufacture "fake news" or manufacture "good reasons" to do this or that. I don't trust Facebook management, but many other "entities" certainly can justify actions based on intentionally misdirected ML usage.

No ethics issues with 3D structure prediction of proteins with AlphaFold: https://alphafold.ebi.ac.uk

---------- END OF DAY 2
---

# Day 1

**Sign-in:**
*Name, Program/Department/Organization, Describe your work/research in a couple of sentences*

- Sarah Stevens, Data Science Hub, I help reserachers learn and use data science/research computing skills
- Chris Endemann, Data Science Hub, I help researchers apply data science and machine learning tools to their research projects - **instructor**
- Scott Wildman, Academic Planning and Institutional Research - HELPER
- Ari Smith, PhD student at Dept. Industrial and Systems Engineering, I am the TA for "Machine Learning in Action for Industrial Engineers" - HELPER
- Trisha Adamus, Ebling Library, I help researchers with many aspects of their data - Helper
- Siddharth Uppal - Pharmaceutical Sciences - Discover natural products from bacteria that can't be cultured in the lab
- Chris Kirby - GLAS Education -
- Meredith Shapiro, Enterprise Claims Data and Analytics at American Family Insurance, I analyze data and create dashboards to help managers judge performance of their teams, claim handling turnaround time etc
- Erwin Lares - Spanish & Portuguese - I work with linguistic data (voice frequencies, duration, intensity, grammatical judgment data). I'll like to learn about way to cluster data and recognize patterns
- Russell DImond, Social Science Computing Cooperative, I consult with social science researchers on data and statistics
- Lauren Baker, Scientist in the School of Veterinary Medicine, Complex genetics in the dog model--currently studying heritable kidney stones and looking to use ML to improve phenotyping
- Liliana Fadul, Research Associate Animal and Dairy Science
- Bethany Moore, Post doc research associate in the Botany department. Currently studying how the lignin pathway has evolved in plants and have previously used ML to predict gene function, but would like to learn more about mearual networks.
- Casey Schacher, Science & Engineering Libraries - HELPER
- James Crall, Assistant Professor in Entomology (CALS), interesting in applying computer vision and deep learning to insect behavior and ecology
- Steve Goldstein, Data Scientist and Project Manager, Data Science Institute; areas of interest: stochastic  processes; computational biology - HELPER

- Jean-Yves Sgro, Biotech center and Biochem Dept.

In our shared etherpad, please answer the following:
1. Where have I seen machine learning in use?
2. What kind of input data does that machine learning system use to make predictions/classifications?
3. Is there any evidence that your interaction with the system contributes to further training?
4. Do you have any examples of the system failing?

Lauren:
1. I have used classification algorithms with genetic (SNP) data and clinical data to try to predict whether an individual will or will not develop a disorder
2. I think it can take almost anything? Anything you would use to make a decision
3. I'm not sure what this means--maybe because I know that I have to be very careful with the variables that I choose to feed into the model
4. the models from 1 weren't very good! We were hoping to get more information from the SNP data, but the SNP data wasn't that informative. The predictions werent great and mostly relied on clinical info.

Sidd:
1. Using Alexa
2. Human voice
3. Yes, it sends information back to amazon
4. Voice recognition fails at times and the training dataset does not cover all questions

Erwin:
1. Siri's recommendations when I get in my car
2. my location
3. when I leave work but don't head home on certain days
4. yup, the one time Siri wanted me to drive to Mexico City because ???

Russell
1. Netflix recommendations
2. Shows watched/not watched
3. Recommendations similar to shows watched recently
4. Sometimes makes recommendations that are not of interest, often becomes "too narrow" and won't show anything that isn't similar to recent shows watched

James
1. Behavioral classification in animals
2. Videos/images of animals
3. Not in this case
4. Plenty! Obviously misclassified behavioral states or improperly identified/tracked animals

Casey
1. Just read this article about ML learning and increased employee surveillance with Microsoft 365: https://winbuzzer.com/2021/11/08/microsoft-365-to-add-increased-employee-surveillance-through-microsoft-edge-xcxwbn/

2. User actions
3. Yep!
4. It is a major privacy fail IMO

1. youtube video closed captions.
2. vast amount of data
3. Yes, over time.
4. foreign language translation

$ cd intro_ML
$ jupyter notebook

Open new jupter notebook with python 3
Name notebook regression instead of untitled

# least squares function - takes data with x and y vectors and will calculate the optimal slope and y intercept

```
def least_squares(data):
    x_sum = 0
    y_sum = 0
    x_sq_sum = 0
    xy_sum = 0

    # the list of data should have two equal length columns
    assert len(data[0]) == len(data[1])
    assert len(data) == 2

    n = len(data[0])
    # least squares regression calculation
    for i in range(0, n):
        x = int(data[0][i])
        y = data[1][i]
        x_sum = x_sum + x
        y_sum = y_sum + y
        x_sq_sum = x_sq_sum + (x**2)
        xy_sum = xy_sum + (x*y)

    m = ((n * xy_sum) - (x_sum * y_sum))
    m = m / ((n * x_sq_sum) - (x_sum ** 2))
    c = (y_sum - m * x_sum) / n

    print("Results of linear regression:")
    print("x_sum=", x_sum, "y_sum=", y_sum, "x_sq_sum=", x_sq_sum, "xy_sum=",
          xy_sum)
    print("m=", m, "c=", c)

    return m, c
```

```
x_data = [2, 3, 5, 7, 9]
y_data = [4, 5, 7, 10, 15]

import matplotlib.pyplot as plt
ax = plt.axes()
ax.scatter(x_data, y_data)

m, c = least_squares([x_data, y_data])


def make_linear(x_data, m, c):
    linear_preds = []
    for x in x_data:
        y = m * x + c
        # add the result to our linear_data list
        linear_preds.append(y)
    return linear_preds

y_preds = make_linear(x_data, m, c)
y_preds

def make_graph(x_data, y_data, linear_preds):
    plt.plot(x_data, y_data, label = "Original data")
    plt.plot(x_data, linear_preds, label = "Line of best fit")
    plt.grid()
    plt.legend()
    plt.show()


make_graph(x_data, y_data, y_preds)

# Calculating RMS (root mean squared) to allow us to assess the model's error quantitatively
# RMS is the sum of squared errors divided by the number of points and then the square root of that
import math

def measure_error(data, preds):
    assert len(data) == len(preds)
    err_total = 0
    for i in range(0, len(data)):
        err_total = err_total + (data[i] - preds[i])**2
    err = math.sqrt(err_total / len(data))
    return err

rms = measure_error(y_data, y_preds)
print('On average, the difference between the model and the real values is:', rms)
```

## Predicting life expectancy

Now lets try and model some real data with linear regression. We'll use the Gapminder Foundation's life expectancy data for this. Download data: https://carpentries-incubator.github.io/machine-learning-novice-sklearn/data/gapminder-life-expectancy.csv

```
# put this line at the top of the file
import pandas as pd

def process_life_expectancy_data(filepath, country, min_date, max_date):

    df = pd.read_csv(filepath, index_col="Life expectancy")

    # get the life expectancy for the specified country/dates
    # we have to convert the dates to strings as pandas treats them that way
    life_expectancy = df.loc[country, str(min_date):str(max_date)]

    # create a list with the numerical range of min_date to max_date
    # we could use the index of life_expectancy but it will be a string
    # we need numerical data
    x_data = list(range(min_date, max_date + 1))

    # calculate line of best fit
    m, c = least_squares([x_data, life_expectancy])
    linear_data = make_linear(x_data, m, c)

    error = measure_error(life_expectancy, linear_data)
    print("error = ", error)

    make_graph(x_data, life_expectancy, linear_data)

process_life_expectancy_data("data/gapminder-life-expectancy.csv",
                             "United Kingdom", 1950, 2010)
```

**CHALLENGE:**
Calculate the life expectancy for Germany between 1950 and 2000. What are the values (m and c) of linear equation linking date and life expectancy?
**ANSWER:**

```
process_life_expectancy_data("../data/gapminder-life-expectancy.csv", "Germany",
1950, 2000)
```

**CHALLENGE:**
Use the linear equation you've just created to predict life expectancy in Germany for every year between 2001 and 2016. How accurate are your answers? If you worked for a pension scheme would you trust your answers to predict the future costs for paying pensioners?
**ANSWER:**

```
for x in range(2001,2017):
    print(x,0.212219909502 * x - 346.784909502)
```

**COMPARE WITH REAL VALUES:**

```
df = pd.read_csv('data/gapminder-life-expectancy.csv',index_col="Life
expectancy")
for x in range(2001,2017):
    y_pred = 0.215621719457 * x - 351.935837103
    y = df.loc['Germany', str(x)]
    print(x, "Predicted", y_pred, "Real", y, "Difference", y_pred-y)
```

# Logarithmic Regression

We've now seen how we can use linear regression to make a simple model and use that to predict values, but what do we do when the relationship between the data isn't linear? As an example lets take the relationship between income (GDP per Capita) and life expectancy. The gapminder website will graph this for us:
https://www.gapminder.org/tools/#$state$time$value=2017&showForecast:true&delay:206.4516129032258;&entities$filter$;&dim=geo;&marker$axis_x$which=life_expectancy_years&domainMin:null&domainMax:null&zoomedMin:45&zoomedMax:84.17&scaleType=linear&spaceRef:null;&axis_y$which=gdppercapita_us_inflation_adjusted&domainMin:null&domainMax:null&zoomedMin:115.79&zoomedMax:144246.37&spaceRef:null;&size$domainMin:null&domainMax:null&extent@:0.022083333333333333&:0.4083333333333333;;&color$which=world_6region;;;&chart-type=bubbles

Download the GDP data from http://scw-aberystwyth.github.io/machine-learning-novice/data/worldbank-gdp.csv

```
def read_data(gdp_file, life_expectancy_file, year):
    df_gdp = pd.read_csv(gdp_file, index_col="Country Name")

    gdp = df_gdp.loc[:, year]

    df_life_expt = pd.read_csv(life_expectancy_file,
                               index_col="Life expectancy")

    # get the life expectancy for the specified country/dates
    # we have to convert the dates to strings as pandas treats them that way
    life_expectancy = df_life_expt.loc[:, year]

    data = []
    for country in life_expectancy.index:
        if country in gdp.index:
            # exclude any country where data is unknown
            if (math.isnan(life_expectancy[country]) is False) and \
              (math.isnan(gdp[country]) is False):
                data.append((country, life_expectancy[country],
                             gdp[country]))
            else:
                print("Excluding ", country, ",NaN in data (life_exp = ",
                      life_expectancy[country], "gdp=", gdp[country], ")")
        else:
```

```
            print(country, "is not in the GDP country data")

    combined = pd.DataFrame.from_records(data, columns=("Country",
                                    "Life Expectancy", "GDP"))
    combined = combined.set_index("Country")
    # we'll need sorted data for graphing properly later on
    combined = combined.sort_values("Life Expectancy")
    return combined
```

### Processing the data

```
def process_data(gdp_file, life_expectancy_file, year, plotTransformedGDP):
    data = read_data(gdp_file, life_expectancy_file, year)

    gdp = data["GDP"].tolist()
    gdp_log = data["GDP"].apply(math.log).tolist()
    life_exp = data["Life Expectancy"].tolist()

    m, b = least_squares([life_exp, gdp_log])

    # list for logarithmic version
    log_data = []
    # list for raw version
    linear_data = []

    for x in life_exp:
        y_log = m*x + b
        log_data.append(y_log)

        y = math.exp(y_log)
        linear_data.append(y)

    if not plotTransformedGDP:
        make_graph(life_exp, gdp, linear_data)
    else:
        make_graph(life_exp, gdp_log, log_data)

    err = measure_error(linear_data, gdp)
    print("error=", err)
process_data("data/worldbank-gdp.csv",
            "data/gapminder-life-expectancy.csv", "1980", False)
```

# Introducing Scikit Learn

```
import numpy as np
import sklearn.linear_model as skl_lin
import sklearn.metrics as skl_metrics
```

```
def process_life_expectancy_data_sklearn(filepath, country, min_date, max_date):
    df = pd.read_csv(filepath, index_col="Life expectancy")

    # get the life expectancy for the specified country/dates
    # we have to convert the dates to strings as pandas treats that way
    life_expectancy = df.loc[country, str(min_date):str(max_date)]

    # create a list w/ the numerical range of min_date to max_date
    x_data = list(range(min_date, max_date + 1))

    # convert x_data and life_expectancy to numpy arrays
    x_data_arr = np.array(x_data).reshape(-1,1)
    life_exp_arr = np.array(life_expectancy).reshape(-1, 1)

    # calculate line of best fit
#     m, b = least_squares([x_data, life_expectancy])
#     linear_data = make_linear(x_data, m, b)
    regression = skl_lin.LinearRegression().fit(x_data_arr, life_exp_arr)

    m = regression.coef_[0][0]
    b = regression.intercept_[0]

    linear_data = regression.predict(x_data_arr)

#     error = measure_error(life_expectancy, linear_data)
    error = math.sqrt(skl_metrics.mean_squared_error(life_exp_arr, linear_data))
    print("error=", error)

    make_graph(x_data, life_expectancy, linear_data)

filepath = "data/gapminder-life-expectancy.csv"
process_life_expectancy_data_sklearn(filepath, "United Kingdom", 1950, 2010)
```

# Clustering with Scikit Learn

https://carpentries-incubator.github.io/machine-learning-novice-sklearn/04-clustering/index.html

### K-means Clustering

K-means clustering algorithm is a simple clustering algorithm that tries to identify the centre of each cluster.

```
import sklearn.cluster as skl_cluster
from sklearn.datasets import make_blobs
import numpy as np
```

```
data, cluster_id = make_blobs(n_samples = 400, cluster_std = 0.75, centers = 4, random_state = 1)
# centers defines the number of centers/clusters
# random state sets it so you get the same different random samples each time you run this function above

print(np.shape(data))
data

print(np.shape(cluster_id))
cluster_id

Kmean = skl_cluster.KMeans(n_clusters = 4)
Kmean.fit(data)
clusters = Kmean.predict(data)

# plot the result
import matplotlib.pyplot as plt
plt.scatter(data[: , 0], data[: , 1], s = 5, linewidth = 0, c = clusters) # s = size of points, c = color by cluster group

for cluster_x, cluster_y in Kmean.cluster_centers_:
    plt.scatter(cluster_x, cluster_y, s = 100, c = 'r', marker = 'x') # much larger, colored red, and using an x to plot centroids


# go back and adjust the cluster_std to be equal to 2 (instead of 0.75) and re-run
# Can see the data in the clusters is starting to run into one another
# try with cluster_std = 10! Everything really overlaps and looks like 1 cluster

# Q: How do you decide how many clusters there should be?
# A: There are metrics to assess the quality of clusters (not in this workshop) - one common one is the silhouette index, tomorrow we will go over some cluster quality indices
# A: clustering is a heuristic and really depends on your data - what your data looks like, what you might use it for.  FOr exmpale, if you have a sense there are 10 groups, you might do 11 and see if it fits.  If it does fit you look at it, see the interpretation, does it give you what you expect.  If not, iterate on that process and use some of these metrics and visualization to guide that exporation.
```

## Using Silhouette Score To Choose Cluster Number

**# Copy the below code (lines 414 - 535) into jupyter notebook**

```
from sklearn.metrics import silhouette_samples, silhouette_score
from sklearn.datasets import make_blobs
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.cm as cm
```

```
# Generating the sample data from make_blobs
# This particular setting has one distinct cluster and 3 clusters placed close
# together.
X, y = make_blobs(
    n_samples=500,
    n_features=2,
    centers=4,
    cluster_std=.5,
    random_state=1, # For reproducibility
)

range_n_clusters = [2,3,4,5]

for n_clusters in range_n_clusters:
    # Create a subplot with 1 row and 2 columns
    fig, (ax1, ax2) = plt.subplots(1, 2)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X) + (n_clusters + 1) * 10])

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 1 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters, random_state=1)
    cluster_labels = clusterer.fit_predict(X)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(X, cluster_labels)
    print(
        "For n_clusters =",
        n_clusters,
        "The average silhouette_score is :",
        silhouette_avg,
    )

    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X, cluster_labels)
    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = sample_silhouette_values[cluster_labels == i]
```

```
    ith_cluster_silhouette_values.sort()

    size_cluster_i = ith_cluster_silhouette_values.shape[0]
    y_upper = y_lower + size_cluster_i

    color = cm.nipy_spectral(float(i) / n_clusters)
    ax1.fill_betweenx(
        np.arange(y_lower, y_upper),
        0,
        ith_cluster_silhouette_values,
        facecolor=color,
        edgecolor=color,
        alpha=0.7,
    )

    # Label the silhouette plots with their cluster numbers at the middle
    ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

    # Compute the new y_lower for next plot
    y_lower = y_upper + 10  # 10 for the 0 samples

ax1.set_title("The silhouette plot for the various clusters.")
ax1.set_xlabel("The silhouette coefficient values")
ax1.set_ylabel("Cluster label")

# The vertical line for average silhouette score of all the values
ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

ax1.set_yticks([])  # Clear the yaxis labels / ticks
ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])

# 2nd Plot showing the actual clusters formed
colors = cm.nipy_spectral(cluster_labels.astype(float) / n_clusters)
ax2.scatter(
    X[:, 0], X[:, 1], marker=".", s=30, lw=0, alpha=0.7, c=colors, edgecolor="k"
)

# Labeling the clusters
centers = clusterer.cluster_centers_
# Draw white circles at cluster centers
ax2.scatter(
    centers[:, 0],
    centers[:, 1],
    marker="o",
    c="white",
    alpha=1,
    s=200,
    edgecolor="k",
```

```
    )

    for i, c in enumerate(centers):
        ax2.scatter(c[0], c[1], marker="$%d$" % i, alpha=1, s=50, edgecolor="k")

    ax2.set_title("The visualization of the clustered data.")
    ax2.set_xlabel("Feature space for the 1st feature")
    ax2.set_ylabel("Feature space for the 2nd feature")

    plt.suptitle(
        "Silhouette analysis for KMeans clustering on sample data with n_clusters = %d"
        % n_clusters,
        fontsize=14,
        fontweight="bold",
    )

plt.show()
```

Challenge