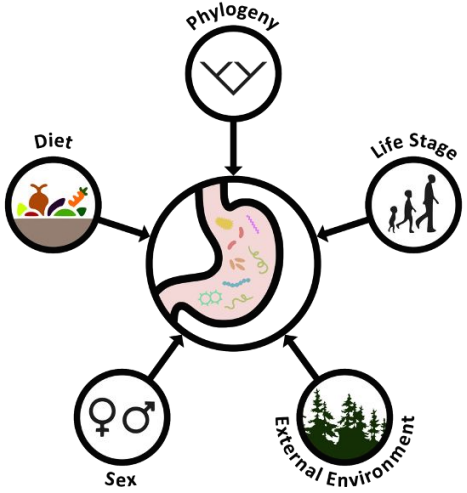
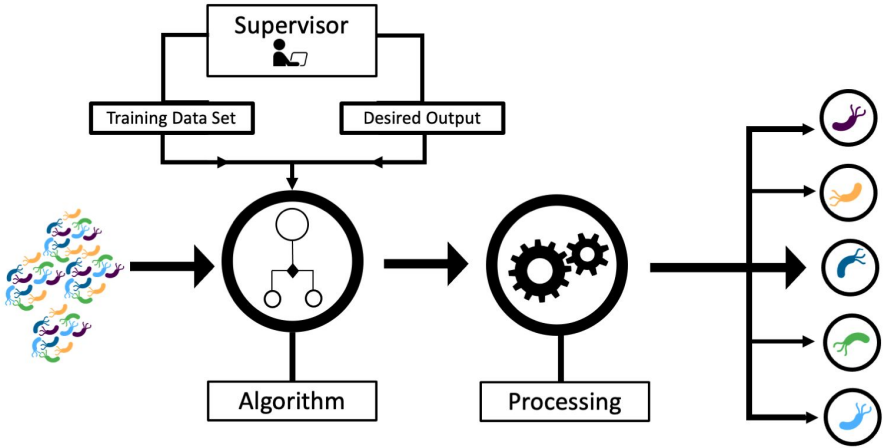
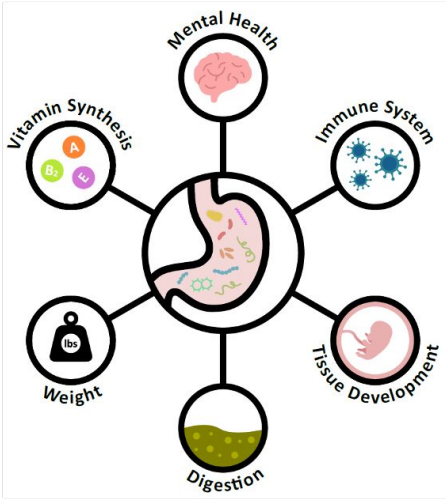
Technology Review:



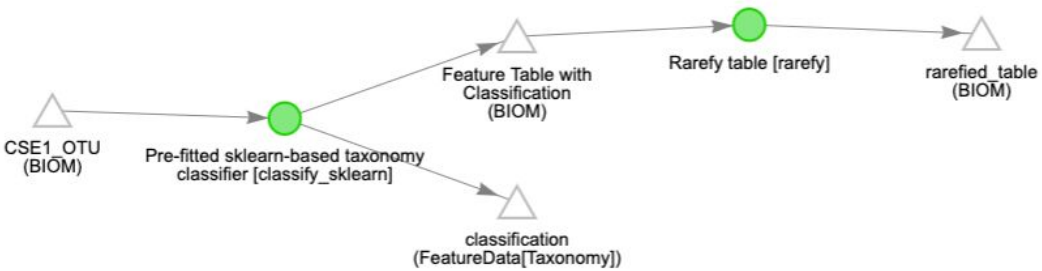
Machine Learning
&
Microbiome





Binnan Yu
Lillian Tatka
Kristina Herman
David Lee
Sierra Gillman

Project background

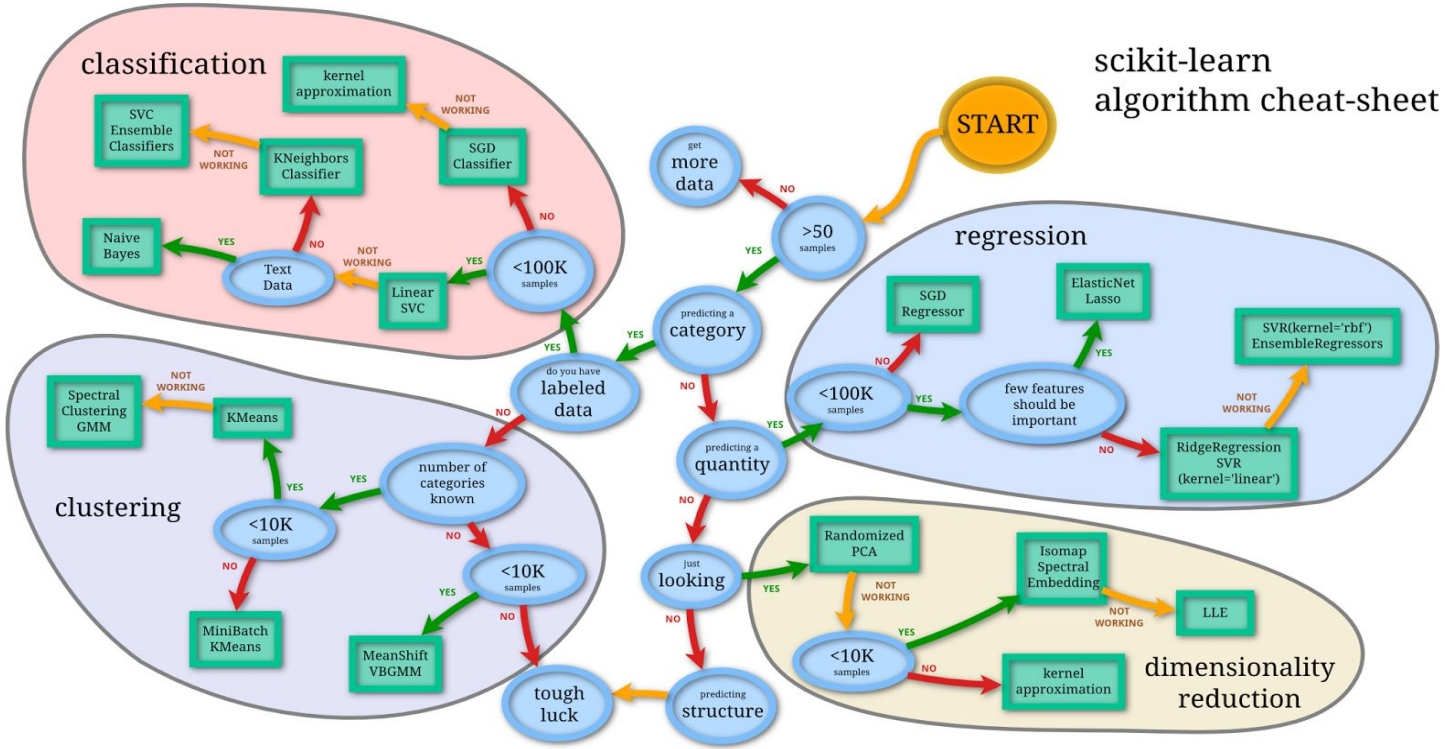


Goal: *Utilize ensemble learning to develop a classification tool that distinguishes subtypes of inflammatory bowel disease (IBD) using fecal microbiome data*

- Looking for packages that implement supervised machine learning algorithms (logistic regression, kNN, decision trees, neural networks, etc.)
- Ideally, also has some cross validation and regularization functions built-in

Scikit-learn Package

1. ML algorithms
2. Hyperparameter optimization



Resource:
https://peekaboo-vision.blogspot.com/2013/01/machine-learning-cheat-sheet-for-scikit.html

- Language that was designed specifically for statistical computing and graphics
- R provides a wide variety of statistical (linear and nonlinear modelling, classical statistical tests, time-series analysis, classification) and graphical techniques.

Advantages: Getting started in R can be easy: built in models & easy to read formula language. R specializes in a smaller subset of statistically-oriented tasks, those tasks tend to be easier to do (at least initially) in R.
Disadvantages: In R, switching between different models usually means learning a new package written by a different author. The interface may be completely different, the documentation may or may not be helpful in learning the package, and the package may or may not be under active development. Python with numpy much faster:

R:

```
> xx <- rep(0, 100000000)
> system.time(xx[] <- 1)
user system elapsed
4.890 1.080 5.977
```

Python:

```
In [1]: import numpy as np
In [2]: xx = np.zeros(100000000)
In [3]: %timeit xx[:] = 1
1 loops, best of 3: 535 ms per loop
```

PyTorch

- Package for implementing **neural networks**, specifically
- Tensor computing (like NumPy)
- Construct networks using nn.Sequential:

```
model = nn.Sequential(nn.Linear(input_size, hidden_sizes[0]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[0], hidden_sizes[1]),
                      nn.ReLU(),
                      nn.Linear(hidden_sizes[1], output_size),
                      nn.Softmax(dim=1))
```

- **Advantages:** Efficient (and easy to use) optimizers, numerous loss functions to choose from, many layer types to choose from (convolutional, pooling, linear, etc.)
- **Disadvantages:** Implement regularization and cross validation “by hand”

Summary of packages

- Easy to learn
- General ML library built on NumPy
- Works great with pandas and matplotlib
- Lower customization
- Great for simple ML algorithms

- More difficult to learn
- More neural-network related utility functions & support
- Higher customization & flexibility
- Suited for deep learning & more complex problems