Comparing scikit-learn functionality to other libraries

Portia Bhattacharjee, Ben Courlang,
Deryus Jijina, Blake Peery

Gaussian Mixture Model

Expectation step: given the parameters of each cluster what is the probability of observing the data in that cluster. P(X|μ,σ^2)

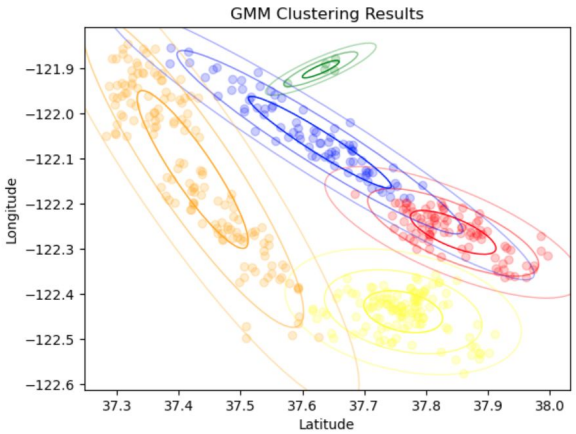
Maximization step: given the share of the data in that cluster what parameters maximize the probability. Max (P(μ,σ^2|X))

Root Finding to find Max(P(μ,σ^2|X))

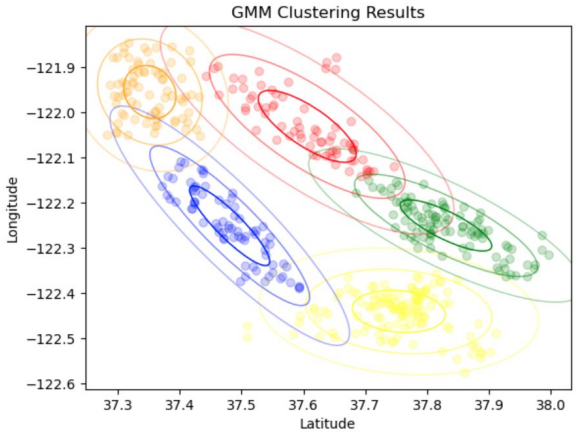
Using 1 for σ^2 or previous value
F = log(P(μ,σ^2|X))
F' = (d/dμ)log(P(μ,σ^2|X))
Find F'=0 using root finding to get max value for μ

Using your found for μ
G = log(P(μ,σ^2|X))
G' = d/dσ^2(log(P(μ,σ^2|X)))
Find G'=0 using root finding to get σ^2

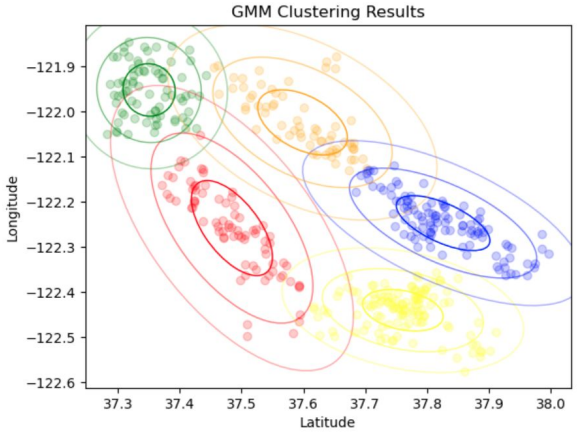
Naive GMM



Sklearn GMM

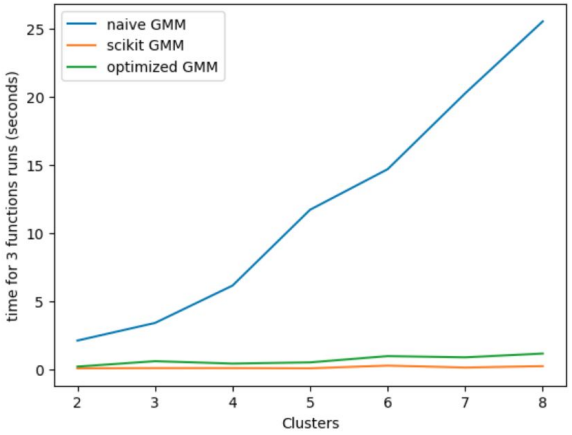


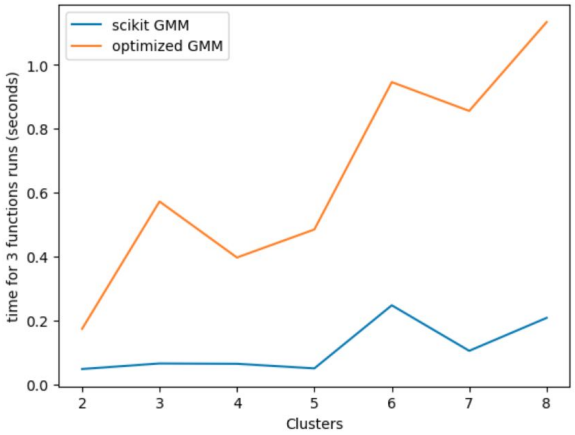
Optimized GMM



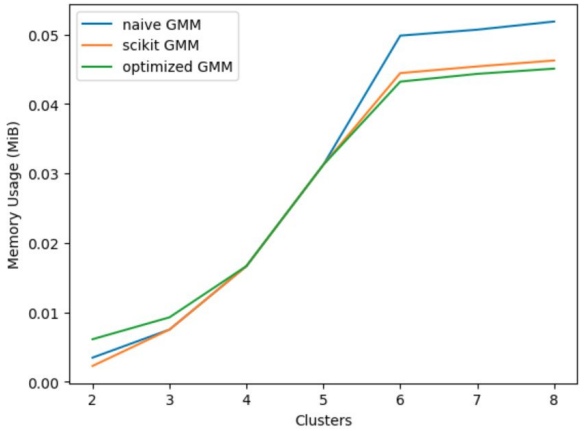
GMM Speed and Memory Usage

Speed





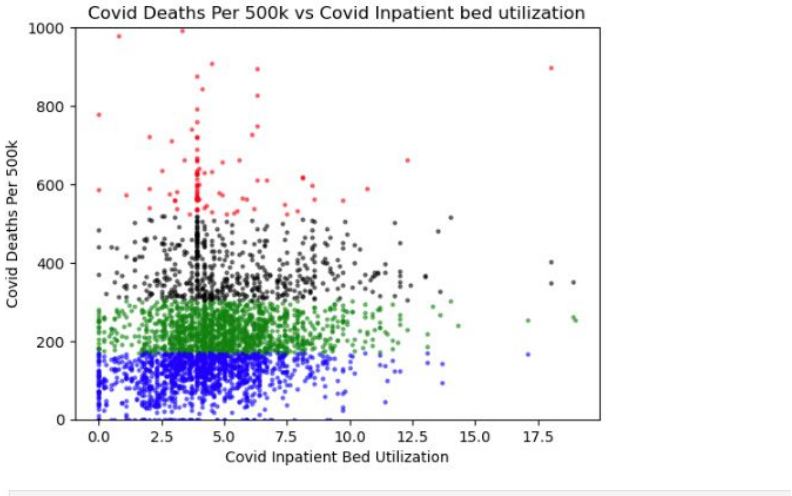
Memory

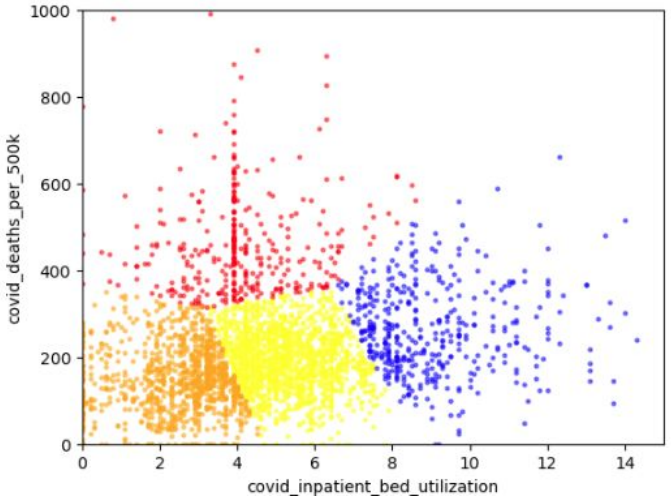


Why the sklearn was the fastest?

They are using the Cholesky decomposition on the covariance matrix to compute an estimation for the log gaussian probability. The Cholesky decomposition is used to find the inverse of the covariance matrix of the sample, which is used to calculate the gaussian probabilities for each data point. In the other GMM algorithms the exact value for each gaussian probability is calculated for each data point. Cholesky decomposition was used because it is stable and well conditioned.

K Means using Covid Data Set





SciKit Learn (Left): Only python library that we could find that has K means implementation, separated clusters vertically for covid data set. Is much faster with execution time of **0.06 seconds**. Used elbow method to determine 4 clusters on dataset.

Coded Algorithm (Right): Algorithm coded by hand finding the centroid for each individual point in the dataframe. Execution time of **31.65 seconds**. Used elbow method to determine 4 clusters on dataset.

Why SciKit was Faster: As stated on SciKit Learn's website: " selects initial cluster centroids using sampling based on an empirical probability distribution of the points' contribution to the overall inertia. This technique speeds up convergence."

Curve Fitting Regression

Scikit

- Bayesian Ridge Regression

average accuracy : 0.7617312999999071

average memory usage : [1294.10968, 22846.16856]

Average time : 4.5 ms

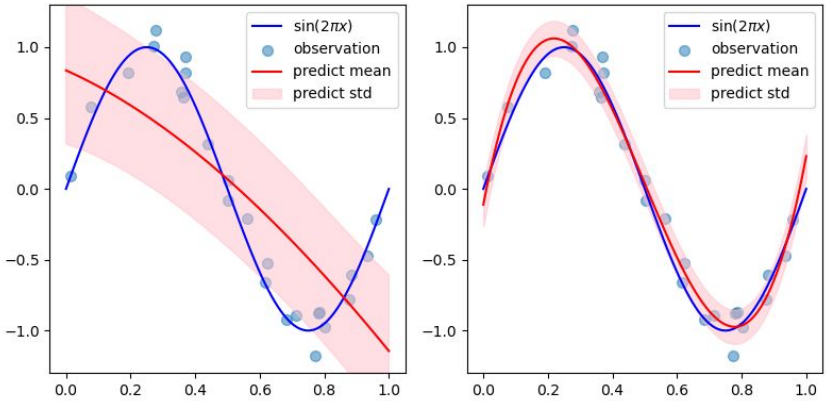
NumPy

- Polyfit

average accuracy : 0.8129321000000155

average memory usage : [407.63824, 9248.07889]

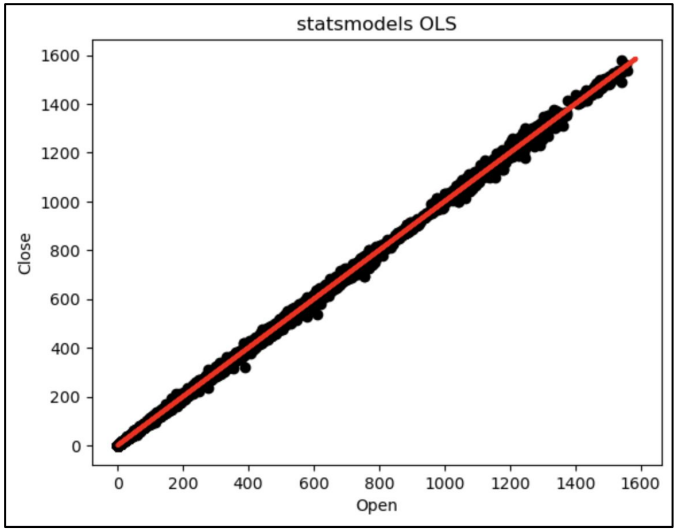
Average time : 0.9 ms



Model Selection

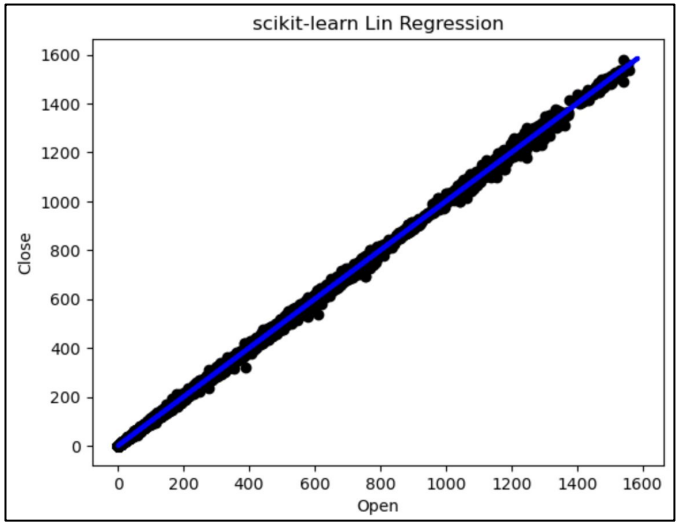
Scikit = better for larger datasets, better preprocessing, most documentation/algos, less advanced stats methods

Statsmodels



Speed: 0.06048226356506348
Mem: 42237597
Accuracy: 0.9996941010180989

Scikit



Speed: 0.046027183532714844
Mem: 1364433
Accuracy: 0.9996941010180989