Brief Overview

The CHEERS project is a software application that uses machine learning and high-performance computing in order to perform material analysis, specifically in this case, determine the level of sugar and alcohol in red wines.

The ML model used is SVR, which looks to set the largest margin rate and lower the misclassification rate, providing the prediction score for any particular target variable (sugar or alcohol).

Without much details, we are going to take only those points which have the least error rate, thus giving us a better fitting model. In order to achieve this, we need to find the optimal combination of the model's free parameters C and epsilon, according to https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVR.html

We want to perform hyperparameter optimization.

The ground truth data for target variables include: Date, Tank, A420, A520, Alcohol, Sugar, Tannins, Inoculation, Color, Varietal.

The parameters of "features" are: Tank, A420, A520, Alcohol, Sugar, Tannins, Inoculation as seen in https://github.com/radical-collaboration/FastFingerPrinting/blob/feature/dataread_per_task/src/galloOSIOPT/code/GalloModel-Parallel.py#L155

What are sugar (and alcohol) search levels? number of processes?

R: Search levels are specified by developers to perform the scan on a deeper level, while number of processes is just as it sounds, defines the number of processes to be used when running the code in parallel according to https://github.com/radical-collaboration/FastFingerPrinting/blob/feature/dataread_per_task/src/galloOSIOPT/code/models.py#L155

Initial Requirements

Functional

1. Which HPO algorithm must be satisfied (grid search, SMBO, K-Means, etc.)?
   - The ML is well defined
   - Training/testing datasets are provided

- Which validation protocol to be used (cross-validation?)
- Parameter search space needs to be defined
- Optimization function needs to be defined (Gaussian process with guided sampling?)

Non-Functional

1. The code must be simple
2. The code must be easy to maintain

Workload Description

It must use the maximum number of cores available on the largest XSEDE machine without significant overhead.

First Approach

We are taking the parameter scan from the code (Chetan's script, specifically getBestSVRMetrics function) and use HyperSpace to convert it to a search since we already have a well defined ML model and provided datasets. We would only need to define the search space according to the free parameters of the model and give it a try.

Solution to be tried using Hyperspace:

- HPO algorithm: SMBO
- ML model is well defined (SVR)
- Training/testing datasets are provided
- Validation protocol to be used is cross-validation
- Hyperparameters search space can be defined (upper and lower bounds can be obtained) from
  https://github.com/radical-collaboration/FastFingerPrinting/blob/feature/dataread_per_task/src/galloOSIOPT/code/models.py#L109
- Optimization function will be Gaussian process with guided sampling

As said before, Chetan's manual grid based search would be a good place to start (https://github.com/radical-collaboration/FastFingerPrinting/blob/feature/dataread_per_task/src/galloOSIOPT/code/models.py#L109). Then, we can also perform a comparison against Andy's Sklearn's GridSearchCV function.

Andy uses the following grid, along with Sklearn's GridSearchCV optimization algorithm:

gsc = GridSearchCV(

```
estimator=SVR(kernel='rbf'),
param_grid={
    'C': [0.1, 1, 100, 1000],
    'epsilon': [0.0001, 0.0005, 0.001, 0.005, 0.01, 0.05, 0.1, 0.5, 1, 5, 10],
    'gamma': [0.0001, 0.001, 0.005, 0.1, 1, 3, 5]
},
cv=9, scoring='neg_mean_squared_error', verbose=1, n_jobs=-1)
```

It could be worthwhile to expand said grid to add more kernels. Andy mentioned that the linear kernel sometimes outperforms the radial basis function. Also the polynomial and sigmoid could be looked at. Chetan’s code already includes these.