**229352 Statistical Learning for Data Science 2** **Fall 2022**

Lab 5: due Saturday August 13

*Instructor: Donlapark Ponnoprat*

**Make sure that you answer all the questions. The report must be turned in as a link to your Colab file.**

# Training support vector machines on text data

Load the `20 newsgroups` dataset with the following code:

```
from sklearn import datasets
train = datasets.fetch_20newsgroups(subset='train')
test = datasets.fetch_20newsgroups(subset='test')
```

1. Transform both training set and test set into tf-idf (TfidfVectorizer).

   This should result in a total of 18806 data points with 130107 features

2. Remember that there are primal form and dual form to the SVM optimization.

   First, train the linear SVM model (LinearSVC) on the training set using the primal form; this can be done by setting `dual=False`.

   Then, train the linear SVM model on the training set using the dual form; this can be done by setting `dual=True`.

3. Which form takes longer to train on this particular dataset? What do you think is the reason? (hint: go back and review the slides on SVM)

4. Now we will study the effects of the hyperparameter $C$ on 1) the test accuracy and 2) the number of support vectors

   Let's simplify things a bit; in the following task **you will train model on the subset of data consisting of only** $y = 0$ **and** $y = 1$.

   Train several **kernel** SVM models (SVC) on the training set with RBF kernel and $C = 0.1, 0.2, \ldots, 0.9, 1.0$. For each value of $C$,

   - compute the test accuracy
   - count the number of support vectors

   Make plots of test accuracies by values of $C$ and numbers of support vectors by values of $C$.

5. What is the best value of $C$?

6. What is the relationship between $C$ and the number of support vectors? Explain this relationship using what we learned from the lecture.

## Train boosted tree models on a simulated dataset

Perform GridSearchCV of the following three models on the provided training set (`X_train.csv` and `y_train.csv`) and evaluate these models on the test set (`X_test.csv` and `y_test.csv`).

For each model, plot the **feature importances** calculated by the model, which can be obtained by calling the library's `plot_importance` function. Here is a minimal example in `XGBoost`:

```
from xgboost import XGBClassifier, plot_importance
model = XGBClassifier()
model.fit(Xtrain, ytrain)
plot_importance(model)
```

- AdaBoost. Grid search over `n_estimators` and `learning_rate`.
- XGBoost. Grid search over `n_estimators`, `max_depth` and `learning_rate`.
- LightGBM. Grid search over `n_estimators`, `max_depth` and `learning_rate`.