Project Definition

COMP 3202

Winter 2025

1 Objective

This project which is **on an individual basis** aims at providing hands-on knowledge of producing, selecting, tuning, and improving a predictive model using `scikit-learn`, which is the most common Python's package for Machine Learning.

The **due date** of the project: **April 7.**

2 Project outline

This project comes with two datasets, *train* and *test*. The train dataset consists of 71 descriptive features and a (binary) target feature, which is the last feature in the dataset. Note that the test dataset (for grading purposes) does not have a target feature. Your *final* model makes a prediction for each data instance in the test dataset. The project involves **three steps:**

1. **Producing bottom-line models**
2. **Tuning the bottom-line models**
3. **Selecting the final model and producing predictions**

3 Project details

It would be a good idea to familiarize yourself with the interface and various modules of `scikit-learn` before starting the project. Here you can find a great resource. Also, before starting your project, make sure that you study Section 4 as well, as the evaluation scheme of this project will inform your activities.

3.1 Producing bottom-line models

In this step you produce 3 bottom-line classifiers, based on *Random Forest*, *k Nearest Neighbor* (k-NN) , and *Support Vector Machine* (SVM) (you may find the `scikit-learn` API of the mentioned classifiers in Random Forest, k-NN, and SVM). Note that each of the mentioned classifiers, includes one or more (hyper-) *parameters*. These parameters can be tuned for better predictive performance (the next step). In this step, however, we focus solely on production of classifiers with *default values*; hence the term "bottom-line". A simple example of training a bottom-line model can be found here.

3.2 Tuning the bottom-line models

This step aims at tuning the bottom-line classifiers based on *Cross Validation* and *F1* measure of predictive performance. This means finding the best values of the tuning parameters of all the classifiers. This is done through *Grid/Random Search* based on Cross Validation implemented in `scikit-learn` as `GridSearchCV` and `RandomizedSearchCV`. You may consult here, here, and here regarding `GridSearchCV`, and `RandomizedSearchCV` and their differences.

You will have to make certain choices to tune your model. Any such choice has either a positive or negative effect on the predictive performance of the models. Note that there is no recipe for making the right choices. The process of making the right choices usually involves trial and error.

1. You need to choose between `GridSearchCV`, and `RandomizedSearchCV`.
2. You need to devise a strategy (the `cv` argument in `GridSearchCV` / `RandomizedSearchCV`). You may use a default strategy devised by `scikit-learn` (note that a default strategy may or may not lead to a decent predictive performance).
3. You need to choose a number of tuning parameters (at least two) for each model.

After doing so, you should end up with three *tuned models*, meaning you have three classifiers based on Random Forest, k-NN, and SVM for each of which you have opted for the parameter values that produce the best predictive performance.

To *improve the predictive performance* of your models, you may also employ *feature engineering* (such as scaling the descriptive features) and/or *feature selection* (such as dropping a descriptive feature) practices. Note however that this is optional (refer to the submission and evaluation section for more information).

3.3 Selecting the final model and producing predictions

Finally, select the best among the three tuned models (in terms of predictive performance) as your final model; and use the final model to predict the target levels of the data instances in the test dataset.

4 Submission and evaluation

You need to submit 4 files: a **Jupyter Notebook,** a **Python script,** a prediction **text file**, and a **PDF** file.

Jupyter Notebook. All your coding should be done and saved in a Jupyter Notebook. Here is an introduction to (IPython and) Jupyter. The platform (local machine, juypter.org, or Google Colab) on which you develop your notebook is up to you.

Ptyhon script. The script file contains your final *trainable* model. More precisely, the script must take two command-line arguments: a filename specifying a CSV file containing the training data, and a filename specifying a CSV file containing the test data. Having the final untrained model with optimized parameter values, your script then uses the training data to train the model, predict the target values of the test data, and output the corresponding *text file* (discussed next). Please make sure that your script runs in the Linux environment.

Text file. You also need to submit the predictions of your model in a separate text file. Where the predicted target values of the instances in the test data appear in a single *column* and *in the same order*.

PDF. Finally, in a PDF file you need to detail and explain all the steps you have taken and all the choices you have made along with the results. Therefore, it might be a good idea to keep a log while working on your project. Needless to say that your submitted Jupyter Notebook must reflect the content of the PDF file.

The **evaluation** of a submitted project consists of the following:

The submitted Jupyter Notebook and the PDF file together are worth 15 points (if they comply with the requirements,, of course).

A working Python script that does what is supposed to (regardless of the quality of its prediction) is worth 2 points.

Finally, the evaluation of predictions will be performed on a competitive basis as follows. The quartiles of F1 scores of all the projects will be calculated. Students whose predictions fall within the 1st quartile receive 1 point out of 3; students whose predictions fall within the 2nd and the 3rd quartile receive 2 points out of 3; students whose predictions fall within the 4th quartile receive 3 points out of 3. Please note that only predictive performance is evaluated; therefore, the mere practice of feature engineering and/or selection does not merit any points on its own.

5 Important notes

Failure to meet the following can cost you full/partial marks.

1. Plagiarism will be easy to spot in this project. If detected, the plagiarist will receive a

grade of 0 (out of 20). Note that the multitude of technical choices embedded in this project makes it extremely unlikely that two submissions resemble each other.

2. Make sure that you submit all the required files.
3. Each student is required to complete this project individually.
4. Project due date will not be extended.
5. The order of the predictions of your model must be the same as the order of the instances in the test dataset.
6. Should you decide to include feature engineering and/or selection, it would be the submitted Python script which performs this as an extra step.

6 Further help

A **Teaching Assistant** is assigned to help you throughout this project. You may email the TA or drop by his office.

Name: Sadman Saumik Islam

Email: `ssislam@mun.ca`

Office Number: EN-1044B

Office Hours:

- Mon: 10 am - 1.00 pm
- Wed: 10 am - 12.00 pm
- Fri: 10 am - 12 pm