CS 181 Practical 1:
Predicting Chemical Toxicity for Drug Design

Kaggle submission closes at 11:59pm on Thursday, February 14
Writeup due 11:59pm on Friday, February 15

You will do this assignment in groups of three. You can seek partners via Piazza. The course staff can also help you find partners. Please see `practical-logistics.pdf` for a description of competing on Kaggle, what to submit to Canvas, and more. Make sure to use the provided LaTeX template for your writeup.

Task description

Determining the chemical toxicity of compounds is a central task in drug design. Use compounds in your drug that are too toxic for biological organisms, and your drug will have unintended adverse effects. Use compounds that aren't toxic enough to a particular kind of bacterium, virus, or mutation, and your drug will be ineffective.

Historically, and for the most part still today, compounds have been manually screened for their toxicity using biological assays. These assays allow us to test for toxicity by letting us observe how much growth in an organism a compound has inhibited. However, manual testing is expensive and time-consuming, and the vast number of compounds one might wish to examine means it is intractable to screen every possible compound in the lab.

A promising line of research therefore focuses on accelerating drug development by using machine learning techniques to predict the toxicity of compounds. These techniques attempt to learn a mapping from feature representations of compounds to a measure of how much these compounds inhibit growth.

Your task is to replicate this work. We have provided you with 5331 compounds to train on, and you are tasked with making predictions for a further 3554. The quantity you are predicting is the percentage of growth the compound inhibits, so note your predictions should fall in the range $[0, 1]$ (the percentage is how much less growth is observed when an organism is treated with the compound than when it is treated with a control).

Just so you know, the features of the compounds we have given you to use for your predictions are molecular descriptors called topological indices. Topological indices are calculated using the molecular graphs of each compound, and so capture each compound's shape and form, as well as the elements it contains. There are 252 features in the data set we have provided.

To see how well your models perform in comparison with those of your classmates, you'll upload your predictions to Kaggle, where a subset of your predictions will be used to produce a public leaderboard. The remaining subset will be used in computing the final rankings at the end of the contest.

Data Files

There are 4 files of interest, which you can get from the Github repository.

- `train.csv` – This file contains information of 5331 compounds in the training set. The first 251 columns are the topological indices. The final column is the percentage of growth inhibition. This is the label you are trying to predict.
- `test.csv` – This file contains information of 3554 compounds in the test set. In this case, the label is not provided. These are the ones you are trying to make predictions for. The first column is a numeric identifier that you should use when constructing your prediction file to upload to Kaggle.
- `sample.ipynb` – This is a simple IPython notebook that produces the predictive files `sample1.csv`. Feel free to build off of this notebook.
- `sample1.csv` – This is an example of a prediction file you might upload to Kaggle. The id numbers match those in the test file, but the predictions are very naïve: just the untuned outputs of linear regression produced by `sample.ipynb`.

Evaluation

After you upload your predictions to Kaggle (which you can do at most four times per day), they will be compared to the held-out targets of the test set, that is the true chemical toxicities of the compounds you have made your predictions for. Your score is computed using the root-mean-square error (RMSE) metric. If there are N test data, where your prediction is $\hat{x}_n$ and the truth is x_n, then the RMSE is

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(\hat{x}_n - x_n)^2}$$

so lower is better.

Sample Code

An IPython notebook is available from the course website to help you get going. The file called `sample.ipynb` is a simple script to load in data and produce a submission to Kaggle with the correct formatting, `sample1.csv`. You don't need to use this file; feel free to build the prediction system as you see fit.

Sample Baseline

On the Kaggle leaderboard, there is a baseline score, generated using Linear Regression without significant parameter tuning. To see how this score was obtained, take a look at `sample.ipynb`. To earn full points, you must achieve a score higher than this baseline. Remember, the public leaderboard will only display a fraction of the *entire* test set, so scores may drastically change if you overfit. After the Kaggle submission closes, the scores on the full test set will be computed and shown. We will be going off the final scores.

Solution Ideas

You have a lot of flexibility in what you might do. You could focus on fancy regression techniques that use the features we provide. We encourage you to use the scikit-learn package from Python when building your models. Here are some ideas to get you started:

- **Ridge Regression:** You could use a simple L_2 regularization approach to regression weights and use cross-validation to determine an appropriate regularization penalty.
- **Lasso Regression:** You could go a bit fancier and use L_1 regression to identify a sparse solution, again using cross-validation to determine an appropriate regularization penalty.
- **Elastic Net:** Use both L_1 and L_2 at once!
- **Neural Network:** Get a jump on the course material and build a neural network to make predictions. Explore the world of deep learning!
- **Ensemble Methods:** Want to be robust? Use bootstrap aggregation with decision tree learning!
- **Support Vector Regression:** Prefer your problem convex? Get a jump on kernel methods and build a support vector machine for regression.
- **Go Totally Bayesian:** Worried that you're not accounting for uncertainty? You could take a fully Bayesian approach to linear regression and marginalize out your uncertainty.
- **Go Nonparametric Bayesian:** Intrigued by the infinite-dimensional machine learning? Check out Gaussian process regression.