Projects

Projects

In the course project, groups of three students will work together to create classifiers for an in-class Kaggle prediction competition. The competition training data is available from the **CS178-F19 Kaggle site** (https://www.kaggle.com/c/uci-cs178-f19). To give your Kaggle account permission to join the in-class competition and upload results, use this URL: **https://www.kaggle.com/t/9a90d98eaed6475cbf4c01685d2245b8** (https://www.kaggle.com/t/9a90d98eaed6475cbf4c01685d2245b8)

After you create your account, you will have the ability to merge your group with your teammates. Please do not do this until *after* Homework 4, as you will want to be able to upload predictors for that homework.

Kaggle Competition

The Problem

Our competition data are customer analytics information, being used to predict (classify) customer behavior like churn (customers leaving the service for another option). There are approximately 7500 data examples provided, consisting of 107 features each. Of these, the first 41 features are numeric (real-valued); the next 28 are discrete categorical, and the final 38 are binary-valued. The nature of the features (what quantities they correspond to) has been either lost or deliberately obscured for privacy reasons. We have also pre-processed the data in a number of ways (balancing the two classes, imputing some missing data, etc.)

Note that this is a relatively small data set, so you will have to balance creating flexible models with regularization or other forms of complexity control, but will not have to worry too much about computational complexity.

The Evaluation

Scoring of predictions is done using **AUC** (https://en.wikipedia.org/wiki/Receiver_operating_characteristic#Area_under_the_curve), the area under the ROC (receiver-operator characteristic) curve. This gives an average of your learner's performance at various levels of sensitivity to positive data. This means that you will likely do better if, instead of simply predicting the target class, you also include your confidence level of that class value, so that the ROC curve can be evaluated at different levels of specificity. To do so, you can report your confidence that it is raining (class +1) as a real number for each test point. Your predictions will then be sorted in order of confidence, and the ROC curve evaluated.

Using Kaggle

Download the training features *X_train*, the training category labels *Y_train*, and the test features *X_test*. You will learn classifiers using the training data, make predictions based on the test features, and **upload your predictions to Kaggle** **(https://www.kaggle.com/c/uci-cs178-f19)** for evaluation. Kaggle will then score your predictions, and report your performance on a random subset of the test data to place your team on the **public leaderboard** **(https://www.kaggle.com/c/uci-cs178-f19/leaderboard)**. After the competition, the score on the remainder of the test data will be used to determine your final standing; this ensures that your scores are not affected by overfitting to the leaderboard data.

Kaggle will limit you to **at most 5 uploads per day**, so you cannot simply upload every possible classifier and check their leaderboard quality. You will need to do your own validation, for example by splitting the training data into multiple folds, to tune the parameters of learning algorithms before uploading predictions for your top models. The competition closes (uploads will no longer be accepted or scored) on **December 11, 2019 at 11:59pm Pacific time** (= 8:00am 12/12 UTC, which is listed on Kaggle).

Submission Format

Your submission must be a file containing two columns separated by a comma. The first column should be the instance number (a positive integer), and the second column is the score for that instance (probability that it equals class +1). The first line of the file should be "Id,Prediction", the name of the two columns. We have released a sample submission file, containing random predictions, named *Y_random.txt*.

Forming a Project Team

Students will work in teams of up to three students to complete the project. We encourage you to start looking for teammates now; one option is to post on Piazza. We require you to finalize your team by November 28.

Please have your team members join an empty "Project Group" under Canvas > People > Groups. This will allow your group to submit one file jointly. (**NOTE:** this is *different* than a "Student Group", which do not work the same or serve the same purpose.) At the **top** of your report, please list your Project Group number, your members (names & IDs), and your Kaggle team name & leaderboard position.

Project Requirements

Each project team will learn several different classifiers for the Kaggle data, as well as an ensemble "blend" of them, to try to predict class labels as accurately as possible. We expect you to experiment with **at least three** (more is good) different types of classification models. While these can be of the same type as used in homeworks, they **must go beyond** the scope of the models used in Homework 4. Suggestions include:

1. **K-Nearest Neighbors.** While the data size for this challenge is not so large, KNN models will need to overcome the relatively high data dimension. As noted in class, distance-based methods often do not work well in high dimensions, so you may need to perform some kind of feature selection process to decide which features are most important. Also, the right "distance" for prediction may not be Euclidean in the original feature scaling (these are raw numbers); you may want to experiment with scaling features differently.
2. **Linear models.** Since you have relatively few input features but a large amount of training data, you will probably need to define non-linear features for top performance, for example using polynomials or radial basis functions.
3. **Kernel methods.** libSVM (https://www.csie.ntu.edu.tw/~cjlin/libsvm/) is one efficient implementation of SVM training algorithms. But like KNN classifiers, SVMs (with non-linear kernels) can be challenging to learn from large datasets, and some data pre-processing or subsampling may be required.
4. **Random forests.** You will explore decision tree classifiers for this data on homework 4, and random forests would be a natural way to improve accuracy. Must go beyond HW4 solutions.
5. **Boosted learners.** Use AdaBoost, gradient boosting, or another boosting algorithm to train a boosted ensemble of some base learner (perceptrons, shallow decision trees, Gaussian naive Bayes models, etc.). Must go beyond HW4 solutions.
6. **Neural networks.** The key to learning a good NN model on these data will be to ensure that your training algorithm does not become trapped in poor local optima. You should monitor its performance across backpropagation iterations on training/validation data, and verify that predictive performance improves to reasonable values. Start with few layers (2-3) and moderate numbers of hidden nodes (100-1000) per layer, and verify improvements over baseline linear models.
7. **Other.** You tell us! Apply another class of learners, or a variant or combination of methods like the above. You can use existing libraries or modify course code. The only requirement is that you understand the model you are applying, and can clearly explain its properties in the project report.

For each learner, you should do enough work to make sure that it achieves "reasonable" performance, with accuracy similar to (or better than) baselines like logistic regression or decision trees. Then, take your best learned models, and combine them using a blending or stacking technique. This could be done via a simple average/vote, or a weighted vote based on another learning algorithm. Feel free to experiment and see what performance gains are possible.

Project Report

By **December 12, 2019**, each team must submit a **single 2-page pdf document** describing your learned classifiers and overall prediction ensemble. Please include:

0. At the **top** of your report, please list your Project Group number, your members (names & IDs), and your Kaggle team name & leaderboard position.

1. A table listing each model, as well as your best blended/stacked model ensembles, and their performance on training and validation and leaderboard data.
2. For each model, a paragraph or two describing: what features you gave it (raw inputs, selected inputs, non-linear feature expansions, etc.); how was it trained (learning algorithm and software source); and key hyperparameter settings (plus your approach to choosing those settings).
3. A paragraph or two describing your overall prediction ensemble: how did you combine the individual models, and why did you pick that technique?
4. A conclusion paragraph highlighting the methods/algorithms that you think worked particularly well for this data, the methods/algorithms that worked poorly, and your hypotheses as to why.

Your project grade will be mostly based on the quality of your written report, and groups whose final prediction accuracy is mediocre may still receive a very high grade, if their results are described and analyzed carefully. But, some additional points will also be given to the teams at the top of the leaderboard.