Alec MacMillen

CAPP30254 – Machine Learning

Assignment #3 – Improving the Pipeline

Due 5/2/2019

**Policy Problem**

Public education suffers from system under-investment in the United States. Many public school teachers do not receive sufficient funding for classroom materials and resources from their school districts, so they are forced to either foot the bill themselves or turn to public fundraising methods like DonorsChoose. DonorsChoose is an online platform on which teachers can post funding requests for “projects,” which can range in type from classroom materials to less tangible learning opportunities like trips. Donors can then view information about these projects (including location, teacher, type of resource and use, number of students reached, etc.) and donate towards the stated funding goal.

The objective of this project is to build and validate machine learning models that will predict which projects posted on DonorsChoose will *not* reach their full funding goals within 60 days of posting. The ultimate aim of the project is to identify 5% of posted projects that are predicted not to reach their funding goals. Possible interventions for these projects include more aggressive advertising on the DonorsChoose website, matched donation incentives, or requests for more teacher input (pictures, descriptions, etc.) to make their requests more attention-grabbing.

**Data Exploration and Pre-processing**

After loading and summarizing the projects\_2012\_2013.csv data, certain variables present themselves as good candidates for features. Namely, the categorical variables *school\_charter, school\_magnet, primary\_focus\_subject, primary\_focus\_area, poverty\_area, grade\_level,* and *eligible\_double\_your\_match* might lend themselves naturally to a decision tree-type classifier that makes decisions using discrete decision rules. Similarly, the *students\_reached* and *total\_price\_including\_optional\_support* continuous variables could be standardized and discretized. Another important step is creating the binary target variable, *not\_funded\_within\_60\_days*, which takes a value of 1 if the project does not reach full funding in 60 days (indicating a need to intervene) and a value of 0 otherwise.

The dataset covers projects posted from 1/1/2012 to 12/31/2013, so there are three six-month periods used to *test* the trained models: 7/1/2012-12/31/2012, 1/1/2013-6/30/2013, and 7/1/2013-12/31/2013. The data from the beginning of the dataset to the day before each of these respective test periods was used to train the models.

For a rough feature selection process, my first step was to convert all potential features to categorical variables, then produce a dataset of all dummy variables and build a series of decision tree classifiers with increasing depth, and then analyzing the most important features of these models. Decision trees will prioritize splitting on attributes that provide the most “information gain” or “purity gain” (depending on the specific criterion selected), which is a handy way of identifying features that carry the most weight in classifying exemplars.

This approach revealed that the 3 most important predictors for decision trees of depth 3-5 were 3 dummy variables indicating whether a given project fell in the 2nd, 3rd or 4th quantile of *total\_price*. Other important features included the project’s eligibility for the “double your impact match”, whether its primary focus area was Math & Science or Literacy & Language, and whether the money donated was being put towards Trips or Technology.

**Model Training and Evaluation**

I trained a variety of different models (K-nearest neighbors, logistic regression, decision trees, support vector machines, random forests, bagging, and boosting) with a variety of different model specifications to determine which were best suited to this task. The two main metrics of interest, in my opinion, are precision and AUC. Given that the confines of the problem dictate that we only have the resources to intervene with 5% of projects, we’re interested in making sure that none of the projects we choose to intervene with are “false positives” that don’t actually need any intervention. Maximizing precision will minimize the false positives we erroneously intervene with. AUC is a useful metric that folds the true positive *and* false positive rates in together, which gives a more holistic overview of how a model performs in predicting which projects won’t get fully funded.

Different models appear to perform best at the thresholds 0.3 and 0.5. The threshold 0.3 maximizes the AUC measure, but the threshold 0.5 maximizes precision. If we’re concerned primarily about avoiding intervening with false-positive projects, a decision-tree classifier using the Gini (impurity) criteria, a random splitter, and a maximum depth of 5 maximizes our precision measure. This is the model I would recommend using.

|  |  |  |  |
| --- | --- | --- | --- |
| **Model & specifications** | **Train/test period** | **Precision** | **AUC** |
| Decision tree  *(criteria=entropy, splitter=random, max\_depth=5, threshold=0.3)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.429 | 0.631 |
| Decision tree  *(criteria=gini, splitter=random, max\_depth=5, threshold=0.3)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.667 | 0.500 |
| K-Nearest Neighbors  *(neighbors=50, metric=Euclidean, weights=uniform, threshold=0.3)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.429 | 0.627 |
| Logistic Regression  *(penalty=L2, C=1.0, threshold=0.3)* | Train: 1/1/12-6/30/13  Test: 7/1/13-12/31/13 | 0.3817 | 0.625 |
| Logistic Regression  *(penalty=L2, C=1.0, threshold=0.5)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.545 | 0.514 |
| Random Forest  *(estimators=10, criteria=Gini, max\_depth=10, threshold=0.3)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.427 | 0.626 |
| Random Forest  *(estimators=10, criteria=Gini, max\_depth=10, threshold=0.5)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.559 | 0.517 |
| Bagging  *(Logistic Regression, estimators=10, threshold=0.3)* | Train: 1/1/12-12/31/12  Test: 1/1/13-6/30/13 | 0.429 | 0.626 |
| Boosting  *(loss=deviance, learning\_rate=0.1, estimators=10, max\_depth=None, threshold=0.3)* | Train: 1/1/12-6/30/13  Test: 7/1/13-12/31/13 | 0.379 | 0.621 |