Building a Student Intervention System

Project 2, Udacity ML Engineer Nanodegree

Abstract

Using machine learning, I created a system to identify students at-risk of failing to enable early targeted intervention.

Mike Cassell

Mike.Cassell@Rogers.com

Classification vs Regression

This project is a classification project since we are attempting to figure out if a student is at risk of failing or not (a binary choice). We are not interested in predicting a specific grade or other number, simply whether they belong to the at-risk group.

Exploring the Data

The requested summary data exploring the dataset were:

Total number of students: 395
Number of students who passed: 265
Number of students who failed: 130
Number of features: 31
Graduation rate of the class: 67.09%

Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem. For each model:

- What are the general applications of this model? What are its strengths and weaknesses?
- Given what you know about the data so far, why did you choose this model to apply?
- Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.
- Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.
-

A Decision Tree Classifier

What is the theoretical O(n) time & space complexity in terms of input size?

According to the scikit-learn reference material, the O(n) time expense for training a tree is $O(n^2 f \log(n))$ where n is the number of samples and f is the number of features. There are alternative implementations in scikit-learn that can be more efficient in some circumstances.

The greatest impact comes from the number of training examples which has a logarithmic effect on the time growth. There is also a linear effect from the number of features (since each node needs to iterate each feature). From what I can see the memory cost should be a constant n plus a small overhead for the nodes as they are computed, there should be no need to copy the training set as subsets can be used through training.

Query time is estimated to be O(log(n)).

What are the general applications of this model? What are its strengths and weaknesses?

Decision trees have a few advantages over other models in that they can handle much less processed data (they can work with categorical values instead of needing dummies) and when not overly complex can create easily understood (white-box) models.

Their downsides are they can tend towards overfitting if given too many features or not structured correctly in training. They can also be biased if the training classes are too uneven and can become computationally expensive in training with too many inputs or features.

Given what you know about the data so far, why did you choose this model to apply?

Our data is limited in both the number of features and observations and has already been cleaned more than necessary for a decision tree so it is a good fit to test with. The added bonus of the model being easily visualized and explained could also prove useful in getting the board's buy in if it is the chosen model.

Result Table

Training Size	Train Time	Training F1	Training Prediction Time	Test Prediction Time	Test F1
100	0.001	0.938776	0	0	0.817518
200	0.001	0.922559	0.001	0	0.788732
300	0.002	0.871194	0	0	0.746269

K Nearest Neighbors

What is the theoretical O(n) time & space complexity in terms of input size?

From our lecture, k Nearest Neighbors tends to have a very low training cost of O(1) in terms of time since it simple loads the data to the model. The memory cost is linear with the size of the training sample or O(n). The trade-off is that the query time is higher since the model needs to find the appropriate point and identify the appropriate points with an O(log(n) + k) although the memory remains constant.

What are the general applications of this model? What are its strengths and weaknesses?

The model is different than Decision Trees since it is instance based and doesn't try and produce a model of the data (instead just polling k training data points that are nearest the queried point to see which class they fit into). Since there is no underlying model, this means that KNN can better represent irregular decision boundaries (since there is no need for a very complex equation to have been modeled) but this is a two edged sword since outliers can skew the boundary or create an area of false positives behind the primary boundary.

Given what you know about the data so far, why did you choose this model to apply?

Scikit's documentation indicates that KNN has been applied in a large number of situations successfully where other models may have difficulty approximating a function to model the data.

Training Size	Train Time	Training F1	Training Prediction Time	Test Prediction Time	Test F1
100	0.016	0.839506	0	0	0.75
200	0	0.843137	0	0	0.751773
300	0	0.824561	0	0.016	0.731343

Linear SVC

What is the theoretical O(n) time & space complexity in terms of input size?

According to scikit-learns documentation, the computational complexity of their implementation is between $O(f \times n^2)$ and $O(f \times n^3)$ depending on the dataset. The choice of kernel could also influence the computational cost of creating the hyperplane. The query cost should be constant since the query is simply being checked against the hyperplane.

What are the general applications of this model? What are its strengths and weaknesses?

Since the model creates a hyperplane that optimizes the separation space between the labeled groups, the groups need to be separable in a feature plane (although it doesn't need to be linear depending on the kernel.)

The model is sensitive to scaled features and so it's important when optimizing to pre-scale them where there is a lot of variation (if one feature is price of a home while another is the square footage, the price would likely have a disproportionate effect since it's on a larger scale.

Given what you know about the data so far, why did you choose this model to apply?

The model works by finding the best dividing plane between the classes and since we are trying to find a dividing plane between the features to predict which students are in danger of failing, it's worth exploring the model.

Training Size	Train Time	Training F1	Training Prediction Time	Test Prediction Time	Test F1
100	0	0.837209	0	0	0.774194
200	0	0.820059	0	0	0.774194
300	0.016	0.811881	0	0	0.774194

Choosing the Best Model

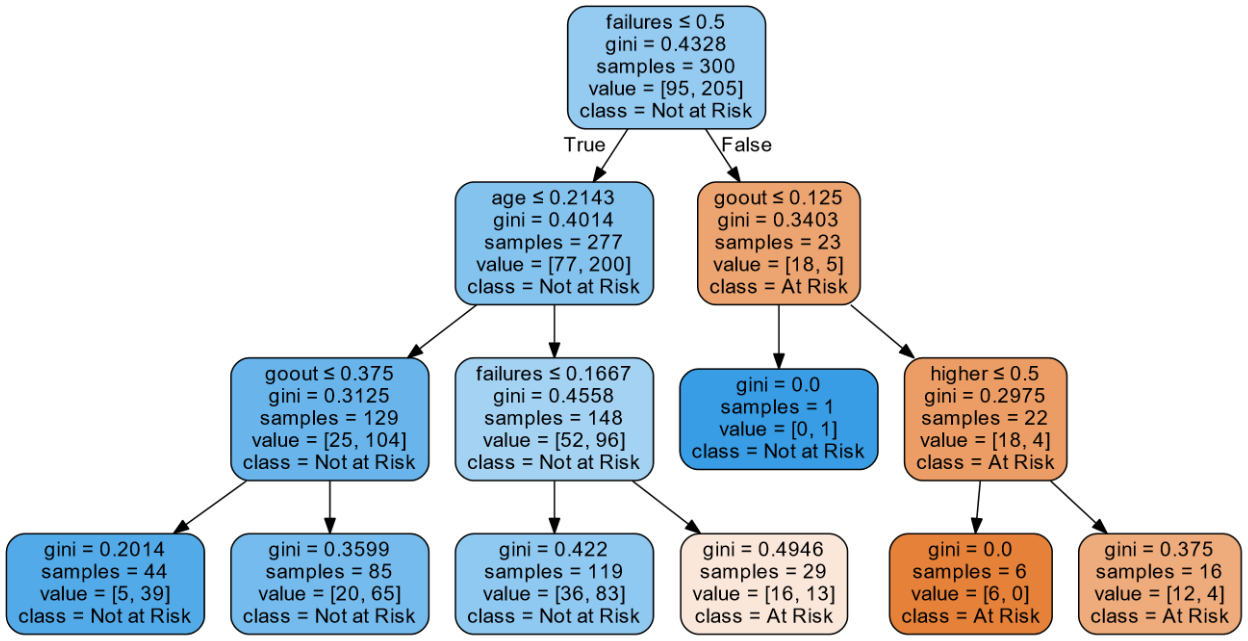
The final model selected was a Decision Tree. It had the optimum F1 score in testing (beating both the KNN and SVM models at all training size levels). The decision tree was average in training time but then the fastest in terms of predictions and creates a small model in memory terms. All models had overall excellent prediction performance on even consumer level hardware with none approaching a 1 second run time but the Decision Tree was slightly faster.

A Decision Tree works by splitting the dataset based on the feature with the highest ability to differentiate between students who are at risk vs. not and splitting the data on that criteria. It then repeats the exercise until it has created a tree of sufficient complexity to differentiate the sample population with an acceptable level of accuracy. As an example, the first split of our tree (illustrated below) is based on if the student's normalized fail rate is above 50%. The students above this mark are then processed separately from those who were below.

As these treatment rules are built, the tree eventually reaches a point where additional splits lead to too small of samples to continue to be accurately generalized to other students. The last node is considered final with the majority of the output labels in it being used as its class (if there are 20 At Risk students and 1 Not at Risk, any student who matches their profile will be considered at-risk.) Once the tree has grown to this point the model is complete and can be used to predict the outcome of newly analyzed students. It does this by applying the same rules it has modeled to the new data points and predicts the final class of the last node.

Through the use of both feature selection and parameter tuning, I increased the F1 score from 74.6% on a training sample of 300 to 84.1% which represents a very accurate model even when taking into account both false positive and negative predictions.

Visualization of Final Model:



References:

Scikit-Learn's documentation: http://scikit-learn.org/

Udacity Discussion Forumns