Building a Student Intervention System

Supervised Learning Project

Template code

Open the template iPython notebook student_intervention.ipynb and follow along.

Project brief

As education has grown to rely more and more on technology, more and more data is available for examination and prediction. Logs of student activities, grades, interactions with teachers and fellow students, and more are now captured through learning management systems like Canvas and Edmodo and available in real time. This is especially true for online classrooms, which are becoming more and more popular even at the middle and high school levels.

Within all levels of education, there exists a push to help increase the likelihood of student success without watering down the education or engaging in behaviors that raise the likelihood of passing metrics without improving the actual underlying learning. Graduation rates are often the criteria of choice for this, and educators and administrators are after new ways to predict success and failure early enough to stage effective interventions, as well as to identify the effectiveness of different interventions.

Toward that end, your goal as a software engineer hired by the local school district is to model the factors that predict how likely a student is to pass their high school final exam. The school district has a goal to reach a 95% graduation rate by the end of the decade by identifying students who need intervention before they drop out of school. You being a clever engineer decide to implement a student intervention system using concepts you learned from supervised machine learning. Instead of buying expensive servers or implementing new data models from the ground up, you reach out to a 3rd party company who can provide you the necessary software libraries and servers to run your software.

However, with limited resources and budgets, the board of supervisors wants you to find the most effective model with the least amount of computation costs (you pay the company by the memory and CPU time you use on their servers). In order to build the intervention software, you first will need to analyze the dataset on students’ performance. Your goal is to choose and develop a model that will predict the likelihood that a given student will pass, thus helping diagnose whether or not an intervention is necessary. Your model must be developed based on a subset of the data that we provide to you, and it will be tested against a subset of the data that is kept hidden from the learning algorithm, in order to test the model’s effectiveness on data outside the training set.

Deliverables

1. Classification vs Regression

This is a classficiation problem. Because the predicted value is 0 or 1. We need to classify students into two categories instead of predicting a numerical value. So obviously, this is a classfication problems.

2. Exploring the Data

By preprocessing the data, we could see these facts:

- Total number of students: 395
- Number of students who passed: 265
- Number of students who failed: 130
- Number of features: 30
- Graduation rate of the class: 67.00%

3. Preparing the Data

```
Feature column(s):-
['school', 'sex', 'age', 'address', 'famsize', 'Pstatus', 'Medu',
 'Fedu', 'Mjob', 'Fjob', 'reason', 'guardian', 'traveltime',
  'studytime', 'failures', 'schoolsup', 'famsup', 'paid',
   'activities', 'nursery', 'higher', 'internet', 'romantic',
    'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health'
, 'absences']

Target column: passed
```

The appendix section contains information on each of these attributes.

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem?

I choose the following four models to analyze the dataset:

- Decision Tree
- Random Forest
- Support Vector Machine

Note: In the ipython notebook decision trees was used. However, it was used merely for a benchmark on f1 score.

Decision Tree

Description:

- Decision tree is a classification algorithms which would by great for classify 0 or 1 problem.

Complexity Analysis:

- It has a space complexity of O(n) which is the worst case, and an average case of O(d^(1/2)*n*log(n)),where n is the number of instances and d is number of instances
- The training time complexity of decision tree model is O(m*nlogn), where n is the number of the number of instances and m is the number of features([1]).

With the figures above, we should see that the expected training and prediction time for NBC should be less due to its linear nature.

Pros:

- It is simple to understand and to interpret. Trees can be visualised[2].

Cons:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning (not currently supported), setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.

- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.[2]

Applications: Decision Tree could be used for financial analysis (especailly for loan decision), natural language processing[3].

Reasons for Selection:

- Decision tree is a good model to predict 0/1 output by splitting at different features, which looks very suitable for this problem.

Training set size	**100**	**200**	**296**
Training time (secs)	0.001	0.005	0.001
Prediction time (secs)	0.000	0.002	0.000
F1 score for training set	1.000	1.000	1.000
F1 score for test set	0.689	0.707	0.629

Thus, for the entire training data, the decision tree will give a F1 score of 0.677 for testing set. So obviously it has a serious overfitting.

Random Forest

Description:

Complexity:

- A Random Forest has a space complexity of O(m^(1/2)*nlog n), where m is the number of features and n the number of instances in the dataset, under the assumption that a reasonably symmetric tree is built.
- For comutational complexity, random forest have two phases, in the buiding phase, the complexity will be O(M*(f)^(1/2)_N log N), where M denotes the number of trees,f means number of features and N means number of instances._
  On the second phase it will be O(MN log N),which means that the overall complexity is dominated by the random forest construction. So overall complexity will be O(M(f)^(1/2)*N log N)[5]

Pros:

- The Random Forests algorithm is a good algorithm to use for complex classification tasks. The main advantage of a Random Forests is that the model created can easily be interrupted.

Cons:

- The main limitation of the Random Forests algorithm is that a large number of trees may make the algorithm slow for real-time prediction.[5]

Applications: -It could be very useful in object detection and regression[6]. It will also be useful in bioinformatics[6].

Reasons for Selection:

- This algorithm was chosen as it is an ensemble of decision tree classifiers, which will dismiss the affection caused by unuseful features.

Training set size	**100**	**200**	**296**
Training time (secs)	0.006	0.012	0.017
Prediction time (secs)	0.001	0.001	0.003
F1 score for training set	0.993	0.993	0.992
F1 score for test set	0.786	0.786	0.701

The classifier was not tuned and default parameters used. Random Forest also have a overfiting issue.

Support Vector Machine

Description:

- In machine learning, support vector machines are supervised learning models with associated learning algorithms that analyze data and recognize patterns. It is very good for 0/1 output learning problems.

Complexity Analysis[7]:

- SVM has a space complexity of O(n^2)
- It has a training time of O(n^3) where n is the training dataset size.

Pros:

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.

Cons:

- It has a high computational cost. Also SVM is also sensitive to noise.

Applications: In industry it will be used for Gene Expression Data Classification; Text Categorization if time permits[7].

Reasons for Selection:

- I expected the data to not be linear with very high dimension. So by using an appropriate kernel like rbf, we would expect to effectively tune the classifier for a great f1 score.

Training set size	**100**	**200**	**296**

Training time (secs)	0.002	0.004	0.007
Prediction time (secs)	0.001	0.002	0.003
F1 score for training set	0.839	0.876	0.880
F1 score for test set	0.765	0.773	0.779

5. Choosing the Best Model

From result of section 4, I would choose Support vector machine provides the best performance.

Overall, Decision tree have a great timing in traning time and extremely high f1 score for training set but very low f1 score for testing set. Random forest is slow in traning but has a good f1 score for testing. SVM also has a similar f1 score for test in compared with Random Forest, however, it is faster.

F1 score for test set is very important because from F1 score for test set, we could see if there is an overfitting problem. Compare with three models, I would first exclude decision tree. It has a very obvious overfitting without having a good f1 score of testing model even though it is very fast.

SVM and Random Forest model have similar F1 score for test set. However, Random Forest have a very high training set so that it is also suspected overfitted. Moreover, though sometimes SVM’s testing f1 score is a slight lower, I would like to trade off with timing.

In conclusion, I would like to pick up SVM in balance with F1 score for test set and training time.

Support Vector Machines are based on the concept of decision lines that define decision boundaries. A decision line is one that separates between different sets of objects. In other words, given labeled training data as is in this supervised learning case, the algorithm outputs a clear divide that categorizes new examples. SVM chooses the best decision line or divide where the distance between that line and the nearest observations of differing classes are the largest.

Furthermore, SVMs can employ the use of kernels. Kernals could be could be understood as specific functions. These kernels or specific functions could fit the data with many features (or so called high demensional data) to convert a linear classifier (a line to classify two categories) into a more complex nonlinear decision (not a line, but maybe a curve) line of both sizes. In this speicic case, the fitted line will predict if a student will graduate or not. Then the line will try to draw a boundary between them.

The chosen SVM model was tuned using Grid Search due to the size of data. Also, in such a case where the data is unbalanced, so I would choose F1 for metric in GridSearch. The parameters optimized were `gamma`, `C` and `tolerance`. And I choose rbf as the kernel. Also I keep f1 score as the metric.

The below is a output from IPython note book which shows the most optimal parameters for SVM. And F1 score is 0.806 for testing data, which shows an improve then section 4.

```
SVC(C=100, cache_size=200, class_weight=None, coef0=0.0, degree=3,
  gamma=0.0001, kernel='rbf', max_iter=-1, probability=False,
  random_state=None, shrinking=True, tol=0.001, verbose=False)
Predicting labels using SVC...
Done!
Prediction time (secs): 0.005
F1 score for training set: 0.891162373815
Predicting labels using SVC...
Done!
Prediction time (secs): 0.001
F1 score for test set: 0.806216530612
```

References:

[1] Utgoff, P. E. (1989). Incremental induction of decision trees. Machine learning, 4(2), 161-186.

[2] scikit-learn documentation for decision tree, http://scikit-learn.org/stable/modules/tree.html

[3] “Applications on decision tree model”, http://www.cbcb.umd.edu/~salzberg/docs/murthy_thesis/survey/node32.ht

[4] http://www.nickgillian.com/wiki/pmwiki.php/GRT/RandomForests

[5] “Random Forest Based Feature Induction”,https://lirias.kuleuven.be/bitstream/123456789/316661/1/ [6] Lecture of Nando de Freitas,https://www.youtube.com/watch?v=zFGPjRPwyFw

[7] Lecture Note of Andrew NG, http://cs229.stanford.edu/notes/cs229-notes3.pdf \

Data

The dataset used in this project is included as `student-data.csv`. This dataset has the following attributes:

- `school` ? student’s school (binary: “GP” or “MS”)
- `sex` ? student’s sex (binary: “F” - female or “M” - male)
- `age` ? student’s age (numeric: from 15 to 22)
- `address` ? student’s home address type (binary: “U” - urban or “R” - rural)
- `famsize` ? family size (binary: “LE3” - less or equal to 3 or “GT3” - greater than 3)
- `Pstatus` ? parent’s cohabitation status (binary: “T” - living together or “A” - apart)
- `Medu` ? mother’s education (numeric: 0 - none, 1 - primary education (4th grade), 2 -€“ 5th to 9th grade, 3 - secondary education or 4 -€“ higher education)
- `Fedu` ? father’s education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 -€“ higher education)

- `Mjob` ? mother’s job (nominal: “teacher”, “health” care related, civil “services” (e.g. administrative or police), “at_home” or “other”)
- `Fjob` ? father’s job (nominal: “teacher”, “health” care related, civil “services” (e.g. administrative or police), “at_home” or “other”)
- `reason` ? reason to choose this school (nominal: close to “home”, school “reputation”, “course” preference or “other”)
- `guardian` ? student’s guardian (nominal: “mother”, “father” or “other”)
- `traveltime` ? home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)
- `studytime` ? weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)
- `failures` ? number of past class failures (numeric: n if 1<=n<3, else 4)
- `schoolsup` ? extra educational support (binary: yes or no)
- `famsup` ? family educational support (binary: yes or no)
- `paid` ? extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)
- `activities` ? extra-curricular activities (binary: yes or no)
- `nursery` ? attended nursery school (binary: yes or no)
- `higher` ? wants to take higher education (binary: yes or no)
- `internet` ? Internet access at home (binary: yes or no)
- `romantic` ? with a romantic relationship (binary: yes or no)
- `famrel` ? quality of family relationships (numeric: from 1 - very bad to 5 - excellent)
- `freetime` ? free time after school (numeric: from 1 - very low to

5 - very high)

- `goout` ? going out with friends (numeric: from 1 - very low to 5 - very high)
- `Dalc` ? workday alcohol consumption (numeric: from 1 - very low to 5 - very high)
- `Walc` ? weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)
- `health` ? current health status (numeric: from 1 - very bad to 5 - very good)
- `absences` ? number of school absences (numeric: from 0 to 93)
- `passed` ? did the student pass the final exam (binary: yes or no)