Machine Learning Engineer Nanodegree

Project 2: Supervised Learning
Building a Student Intervention System

Ayedh Alshahrani

---

02 May 2016

1. Classification vs Regression

The idea of this project is to build system that will help academic decision maker to provide an early intervention to those student who might need it in the right time. The system will use machine learning to analyze the historical records of all students and their final grade (pass/fail). This type of problem is a typical classification on Machine Learning. This is because the response variable or target variable (passed - did the student pass the final exam) has two values (Yes/No) which is a categorical data type or a binary classes. So, the response has a distinct number of values that will be used to classify the problem accordingly. On the other hand, response variable on regression problem will be a continuous numerical variable such as temperature, house's price etc.

Note that in any classification or regression problem the independent variables or predictors values can be categorical or continuous, and only the target's data type do matter here for the type of the problem.

This problem can be converted to a regression in case that the school would like to predict the final grade out of 100 rather than pass or fail. This might be helpful to be suggested to decision maker to provide different level of interventions during the academic year performance. Moreover this might even helpful to the school to change their passing score, for example from 60/100 to 55/100. This type of intervention suggest to change to school system rather than providing help to student directly.

2. Exploring the Data

Table below summarizes the data exploration of the given data:

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	67.09%

The most important note here that the failing rate of the class is about 33% while the graduation rate is about twice that. This is called imbalanced training data. This might cause the learning algorithm to do poorly with the minority class (fail in our case). The performance measure like accuracy might not be useful to deal with such skewed classes. The best measure will be using F1 Score which is a single measure that balances the tradeoff between recall, and precision.

3. Preparing the Data

Data preparation is a critical step in machine learning and we executed different level of preparation as follows:

3.1) Identify feature and target columns

Separating the dataset into features and target will be the first step. Here is the output along with sample of 5 values for each variable as follows:

```
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', 'f
reetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
Target column: passed

Feature values:-
  school sex  age address famsize Pstatus  Medu  Fedu     Mjob      Fjob  \
0     GP   F   18       U     GT3       A     4     4  at_home   teacher
1     GP   F   17       U     GT3       T     1     1  at_home     other
2     GP   F   15       U     LE3       T     1     1  at_home     other
3     GP   F   15       U     GT3       T     4     2   health  services
4     GP   F   16       U     GT3       T     3     3    other     other

    ...    higher internet  romantic  famrel  freetime goout Dalc Walc health  \
0   ...       yes       no        no       4         3     4    1    1      3
1   ...       yes      yes        no       5         3     3    1    1      3
2   ...       yes      yes        no       4         3     2    2    3      3
3   ...       yes      yes       yes       3         2     2    1    1      5
4   ...       yes       no        no       4         3     2    1    2      5

  absences
0        6
1        4
2       10
3        2
4        4

[5 rows x 30 columns]
```

3.2) Preprocess feature columns

As we can see from the sample values, there are many non-numeric values that need to be converted to a format that can be easy to compute by SKlearn APIs. For example (yes/no) can be converted easily to (1/0). Other cateogral variables will be converted into dummy variables, using pandas.get_dummies() . This show the output of this step:

```
Processed feature columns (48):-
['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R', 'address_U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_A', 'Pstat
us_T', 'Medu', 'Fedu', 'Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_teacher', 'Fjob_at_home', 'Fjob_heal
th', 'Fjob_other', 'Fjob_services', 'Fjob_teacher', 'reason_course', 'reason_home', 'reason_other', 'reason_reputation', 'guard
ian_father', 'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activi
ties', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']
```

3.3) Split data into training and test sets

In this step, data will be divided into training and testing using (train_test_split)**,** and the output as below:

```
Training set: 300 samples
Test set: 95 samples
```

4. Training and Evaluating Models

Three classification machine learning algorithms (Decision Trees, Random Forest, and Support Vector Machine) have been selected to model the intervention system. In this section, we will inspect each algorithm and provide details regarding its running time and the impact of training size on the overall performance. Note that, here we try to use the default settings of every algorithm, and this section will be the foundation to choose one or more models, then in section 5, Grid search technique will be used to optimize our selected model and identify the best possible settings of the algorithm parameters.

4.1) Decision Trees Algorithm

Decision Trees is non-parametric supervised learning algorithm can be used for both regression and classification. The idea of the algorithm is to build its prediction of the target variable based on inferring rules out of the input features. The output model is a tree-like model where leaves represents classes while branches represents features' conjunctions. There are different popular implementation of decision trees like: ID3, C4.5, CART (Classification And Regression Trees), and CHAID. The algorithm is top-down approach, and choosing the best split feature is a key. There are different metrics used for choosing the best split such as: Gini impurity, Information gain, and variance reduction. Decision trees are so popular to be used in different domain such as physics, medicine, and software development.

The strength points of this model as follows:

- It is simple to explain, interpret, and visualize in tree-like diagram. So it is easy to communicate with decision makers and to convert the output model into a rules which can be extended and combined manually with Subject Matter Experts rules in any system.
- It can be used as base model to start with. This is because it requires simple preparation and handle both numeric and categorical features. Not only that but also fast to run.

The weakness points of Decision trees are as follows:

- It can build over-complex tree that fit well on training data. This is well known problem called Overfitting.
- It does not handle missing values.

It has been selected to start with because it is easy and simple, and will be used as my base model.

The table below shows the analysis of this algorithm:

	Training set size		
	100	**200`**	**300**
Training time (secs)	0.001	0.001	0.002
Prediction time on training (secs)	0.001	0.001	0.000
Prediction time on testing (secs)	0.000	0.000	0.000
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.644	0.738	0.650

It is clear from the above table that the decision tree overfits the training data and got (1.0) F1 score while got at best about 0.738 on testing when training size is 200.

4.2) Support Vector Machine (SVM)

SVM is one of the powerful machine learning algorithm which can be used for classification, regression and outlier detection. It is used widely in many application on bioinformatics, text, and image recognition. The idea behind SVM is to find the hyperplane/line that will separate or classify data input into different categories. The optimal hyperplane is simply an equation that maximizes the margin of the decision boundary between classes. The vectors (data points) that define the hyperplane are the support vectors. Moreover, SVM extends its capability of linearly separable data to work in nonlinear settings as well using kernel concept. The kernel maps data input/features into higher dimensions that changes the actual data representation, then apply the linear model learning of SVM with the new representation of the features.

Some of the strength points of SVM as follows:

- It handles high dimensional data effectively.
- The support vector, we explained above, built as part of the training data, this means SVM is memory efficient algorithm.
- It has regularization parameters that can be tuned to avoid overfitting problem.
- SVM support the following kind of kernels: linear, polynomial, RBF, and sigmoid. Not only that but also it supports custom based kernels. This show the real power of SVM and how it is flexible and extensible to suit wide range of real world problems in one box.

Some of the weakness point of SVM as follows:

- It is very slow and memory intensive in training and testing compared to others. This can be notices once the training size increased in term of number of instances and number of features.

- Kernel choice is not automatic and need to try different choices.

-

The table below shows the analysis of this algorithm:

	Training set size		
	100	**200`**	**300**
Training time (secs)	0.001	0.004	0.008
Prediction time on training (secs)	0.001	0.003	0.005
Prediction time on testing (secs)	0.001	0.002	0.002
F1 score for training set	0.877	0.867	0.876
F1 score for test set	0.774	0.781	0.783

The table shows the F1 score improved with the increase of the training data. The training time also doubled once we increase the training data, but in our case the full training size is still small and such time is comparable to our decision tree (our base model). F1 score is 0.783 while Decision tree is 0.738. This means SVM still will do better in case we tunes its parameters and we might get better performance. It is also important to notice that the F1 score for the training set for SVM is comparable to the testing F1 score which means that the algorithm in its basic setting generalize better than both our base model and Random Forest.

4.3) Random Forest

Random Forest is ensemble learning approach. Ensembles are a divide-and-conquer used to improve performance. The idea basically to group different week-learners (such as our base model) to form stronger one. Random Forest is an evolution of decision trees where it constructs multiple decision trees during the training time on various sub-samples of training data, then output the mode of the classes (for classification) or mean (for regression) of every individual trees used. Random Forest has been used different real life application such as medicine, bioinformatics, and text processing.

Some of the strongest points of this algorithm:

- It can scale well. Moreover, it can handle missing values and imbalanced classes.
- It is suitable for complex classification task.

Some of the weakness points:

- The output model does not provide the same simplicity and rules as the decision trees.
- Real time prediction might become slow in case of large number of trees used.

The table below shows the analysis of this algorithm:

	Training set size		
	100	**200`**	**300**
Training time (secs)	0.009	0.011	0.010
Prediction time on training (secs)	0.001	0.002	0.001
Prediction time on testing (secs)	0.002	0.001	0.001
F1 score for training set	0.984	0.996	0.990
F1 score for test set	0.677	0.805	0.724

The best performance of Random Forest when training size is 200, this is the same case as in our base model. F1 score is 0.805. The training time is greater than our base model but the prediction time is almost the same as we have in our base model. The most important note here is that the F1 score for training data overfits as our base model did.

5. Choosing the Best Model

To choose the best model to recommend for school decision makers, there are many factors need to be taken in consideration. Table below summarizes the scores of the three chosen algorithms based on our analysis in section 4:

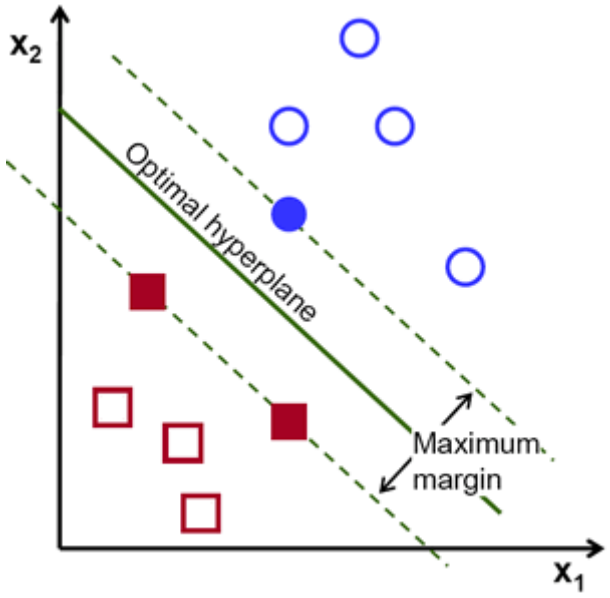
Model	**Decision Trees**	**Random Forest**	**SVM**
Training size	200	200	300
Training time in secs	0.001	0.011	0.008
F1 Score for training	1.0	0.996	0.876
F1 Score for testing	0738	0.805	0.783

The first factor is running time for the training as well as for prediction. This is very important in case we moved into production environment. Since the data size in hand is relatively small, it is about 300 records, the time factor become less important and comparable between all. For example the training time for all algorithms: Decision trees (0.001), Random Forest (0.011), and SVM (0.008).

The second factor is the testing score of F1. It is obvious that both Random Forest (0.805), and SVM (0.783) are better than our base model (0.738) (Decision Trees). However, the training score for our base model and for Random Forest shows a clear fitting for the training data (overfitting problem). The tradeoff between the running time and overall performance, SVM is the model that will be chosen for building our intervention system.

The idea behind SVM as explained earlier in section (4.2) is to find the hyperplane/line that will separate or classify data input into different categories. The optimal hyperplane is simply an equation that maximizes the margin of the decision boundary between classes. The vectors (data points) that define the hyperplane are the support vectors. The diagram below shows the optimal hyperplane of SVM. ***This diagram has been taken***

from OpenCV (*Open Source Computer Vision Library****) website, an open source for library for machine leering.***



Moreover, SVM extends its capability for non-learner problems by employing kernel trick. The kernel basically maps data input/features into higher dimensions that changes the actual data representation, then apply the linear model learning of SVM with the new representation of the features.

For our final model, Grid Search has been implemented to search for the best settings for SVM. It will search for three important parameters: kernel, C, and gamma.

Based on SKlearn documentation: "*Intuitively, the gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The gamma parameters can be seen as the inverse of the radius of influence of samples selected by the model as support vectors. The C parameter trades off misclassification of training examples against simplicity of the decision surface. A low C makes the decision surface smooth, while a high C aims at classifying all training examples correctly by giving the model freedom to select more samples as support vectors*".

Stratified Shuffle Split has been used instead of cross-validation with Grid Search because the training data set is only about 300 records, not only that but also classes are unbalanced as concluded earlier in section 2.

The final model settings as follows:

```
SVC(C=800, cache_size=200, class_weight=None, coef0=0.0, degree=3,
  gamma=1e-05, kernel='rbf', max_iter=-1, probability=False,
  random_state=None, shrinking=True, tol=0.001, verbose=False)
```

The performance of the chosen model as follows:

```
Predicting labels using SVC...
Done!
Prediction time (secs): 0.004
F1 score for training set: 0.831275720165
Predicting labels using SVC...
Done!
Prediction time (secs): 0.002
F1 score for test set: 0.794520547945
```

The final result is 0.794 which is an improvement compared to the default settings for SVM which is (0.78).