# P2: Building an Intervention System

## *Classification vs Regression*

**Your goal is to identify students who might need early intervention - which type of supervised machine learning problem is this, classification or regression? Why?**

This is a classification problem as we wish to assign a class (or label) from a finite set of classes to an observation, ie. the target variable is categorical. In this example we are wanting to predict the value of the target variable or class 'passed' which indicates whether or not the student passed the final exam. The class 'passed' has a finite set of values or categories it can assume, specifically, either 'yes' or 'no'.

Contrast this with a regression problem where the target variable is continuous.

## *Exploring the Data*

**Can you find out the following facts about the dataset?**

| | |
|---|---|
| Total number of Students | 395 |
| Number of students who passed | 265 |
| Number of students who failed | 130 |
| Graduation rate of the class (%age) | 67.09% |
| Number of features | 30 |

## *Preparing the Data*

**Execute the following steps to prepare the data for modeling, training and testing:**

- **Identify feature and target columns**
- **Preprocess feature columns**
- **Split data into training and test sets**

**Starter code snippets for these steps have been provided in the template.**

I used sklearn's train_test_split to create a training set of 301 samples (76%) and a test set of 94 samples (24%).

## *Training and Evaluating Models*

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

I have chosen the following three supervised learning models:

1. DecisionTreeClassifier
2. Support Vector Machine
3. GradientBoostingClassifier

## DecisionTreeClassifier

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

As the DecisionTreeClassifier model can be used for both Classification and Regression problems, it has many applications, some of these include: text analysis, pattern recognition, medical diagnostics, face and speech recognition, power system stability predictions, noise filtering from images, financial analysis and prediction etc.

### *Strengths*

The DecisionTreeClassifier model has the following strengths:

- They are easy to use and interpret as the trained tree can be graphically visualised.
- The model can handle both numerical and categorical data.
- Decision trees require little effort in terms of data preparation such as normalisation or scaling as the tree structure will remain the same with or without the transformation.
- Decision trees are robust against skewed distributions and outliers as assumptions on variable distribution are not made when constructing axis splits.
- Decision trees are considered 'white boxes' in that their acquired knowledge can be expressed in a readable format, where as a model such as an SVM is a 'black box' model as its acquired knowledge cannot be read in a comprehensible way.
- Decision trees are non-parametric, they don't make assumptions about the data distribution, so you don't have to worry about whether the data is linearly separable or not.

### *Weaknesses*

The DecisionTreeClassifier model has the following weaknesses:

- Without proper pruning or limiting tree growth decision trees tend to overfit the training data which leads to poor predictions on the test data.
- Finding the provably smallest tree is an NP-Hard problem, 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.

- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. Again this can be mitigated by training multiple trees in an ensemble learner.
- Concepts such as the XOR and parity problems are hard to learn because no individual attribute exhibits any significant association to the class, and structure only becomes apparent in the fully expanded tree. However these types of problems are rare in practice.

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

I chose the models I did primarily to demonstrate a range of different types of supervised learning algorithm's, from a simple model to more complex model types.

The DecisionTreeClassifier was chosen as most of the input data features, especially after some minor pre-processing, were binary in nature which are ideal for decision trees.

**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.**

| | Training Set Size | | |
|---|---|---|---|
| | 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.002 | 0.003 |
| Prediction time (secs) | 0.001 | 0.000 | 0.000 |
| F1 score for training set | 1.0 | 1.0 | 1.0 |
| F1 score for test set | 0.72 | 0.746031746032 | 0.730158730159 |

# Support Vector Machine

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

As the Support Vector Machine model can be used for both Classification and Regression problems, it has many applications, some of these include: text analysis, pattern recognition, medical diagnostics, electric load forecasting, face and speech recognition, bioinformatics (gene expression and protein sequencing), machine vision and time series analysis etc.

### *Strengths*

The Support Vector Machine (SVM) model has the following strengths:

- SVM's tend to generalise well as once a hyperplane is found, most of the data other than the support vectors (points closest to the boundary) become redundant. This means that small changes to data cannot greatly affect the hyperplane and hence the SVM.
- Based upon sound mathematical theory ie. Developed from statistical learning theory.
- SVM's are particularly well suited to types of data where the number of features can be very large compared to the number of data points.
- Using a non-linear kernel can capture much more complex relationships between datapoints without having to perform difficult transformations myself.
- Few parameters to consider and tune. ie. Kernel, C
- Robust to outliers, the margin parameter C is used to control the misclassification error. Tuning this parameter can suppress outliers by allowing them to be misclassified if desired.
- Cannot be trapped in a local minima, as an SVM is formalised as a quadratic programming problem there will be a global optimum solution.
- Works well with fewer training samples as the number of support vectors do not matter much.
- Uses the kernel trick to find non-linear solutions efficiently.
- Handles data that is not linearly separable.

### *Weaknesses*

The Support Vector Machine (SVM) model has the following weaknesses:

- One of the SVM's strengths leads to one of its biggest drawbacks in that the complex data transformations and resulting boundary plane are very difficult to interpret intuitively. Hence SVM's are known as a 'Black Box', this makes it difficult to incorporate domain knowledge.
- Training/learning time can take a long time as its much more computationally intensive due to the quadratic programming optimisation required.

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

I choose the models I did primarily to demonstrate a range of different types of supervised learning algorithm's, using a simple model to more complex model types.

The SupportVectorMachine model was chosen due to the nature of the data., specifically the dataset has a reasonably high number of features (after pre-processing) relative to the small amount of data points.

**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.**

| | Training Set Size | | |
|---|---|---|---|
| | 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.004 | 0.010 |
| Prediction time (secs) | 0.001 | 0.001 | 0.002 |
| F1 score for training set | 0.869565217391 | 0.872274143302 | 0.869379014989 |
| F1 score for test set | 0.784615384615 | 0.78125 | 0.777777777778 |

## GradientBoostingClassifier

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

As the GradientBoostingClassifier model can be used for both Classification and Regression problems, it has many applications, some of these include: web search ranking, neurorobotics, text analysis, pattern recognition, medical diagnostics, face and speech recognition, power system stability predictions, noise filtering from images, financial analysis and prediction etc.

### *Strengths*

The GradientBoostingClassifier model has the following strengths:

- Tree ensembles given they are nothing more than a bunch of decision trees combined can handle categorical and binary features very well.
- Tree ensembles do not expect linear features or even features that interact linearly, in fact they can automatically detect non-linear feature interactions.
- Reduces variance as results are less dependent on features of a single model and training set.
- They have support for different loss functions.
- They can provide a measure indicating how important or useful a variable is in predicting the outcome.
- Works well with heterogeneous data ie. features can have different scale. In fact tree ensembles inherit the natural data handling properties of decision trees.

### *Weaknesses*

The GradientBoostingClassifier model has the following weaknesses:

- As a GradientBoostingClassifier uses decision tree's as weak learners it can inherit the weaknesses of the weak learner. See section on DecisionTreeClassifier weaknesses for details.
- Gradient boosting models can comprise hundreds or thousands of decision trees thus they cannot be easily interpreted by visual inspection of the individual trees.
- The GradientBoostingClassifier has many parameters that can be specified and thus requires careful tuning.
- They are slow to train as there can be many hundreds or thousands of decision trees to build

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

I choose the models I did primarily to demonstrate a range of different types of supervised learning algorithm's, using a simple model to more complex model types.

The GradientBoostingClassifier model was chosen as I wanted to include an ensemble learner.

Ensemble learners generate multiple hypotheses using the same base learner to obtain better prediction performance than could otherwise be obtained from a constituent component of the ensemble.

**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.**

| | Training Set Size | | |
|---|---|---|---|
| | 100 | 200 | 300 |
| Training time (secs) | 0.078 | 0.116 | 0.155 |
| Prediction time (secs) | 0.000 | 0.010 | 0.000 |
| F1 score for training set | 1.0 | 0.992907801418 | 0.971428571429 |
| F1 score for test set | 0.760330578512 | 0.775862068966 | 0.789473684211 |

## *Choosing the Best Model*

**Based on the experiments you performed earlier, in 1-2 paragraphs explain to the board of supervisors what single model you chose as the best model.**

Based on my experiments of building and training three supervised learning models (all with default parameters), specifically a DecisionTreeClassifier, a Support Vector Machine and a GradientBoostingClassifier, I found that the GradientBoostingClassifier gave the best prediction accuracy on the supplied data.

I found that the GradientBoostingClassifier out performed on accuracy when compared to the DecisionTreeClassifier on all sizes of training data, as did the Support Vector Machine. The Support Vector Machine's accuracy was very consistent across all training set sizes, including the full training set, and was better than the GradientBoostingClassifier on training set sizes of 100 and 200. The GradientBoostingClassifier did outperform the SVM on the training set size of 300.

When trained on the full set of training data the following F1 scores were observed (from the supplied notebook code):

| **Model** | **F1 Score for test set after training on full training set** |
|---|---|
| DecisionTreeClassifier | 0.730158730159 |
| Support Vector Machine | 0.787401574803 |
| GradientBoostingClassifier | 0.820512820513 |

This provides further evidence that the GradientBoostingClassifier is the best choice as its accuracy when trained on the full training set is significantly better when compared to the other two models.

The DecisionTreeClassifier and Support Vector Machine did out perform the GradientBoostingClassifier when comparing the training times, by at least 10x times. However, as all training times were less than 1 second this is not significant. This could become significant if the amount of training data was to dramatically increase.

So given that the goal is to model the factors that predict how likely a student is to pass their high school

final exam, I have chosen model GradientBoostingClassifier as it gives the most accurate predictions and with its large number of parameters offers the best chance to increase the prediction accuracy with parameter tuning.

**Which model is generally the most appropriate based on the available data, limited resources, cost, and performance?**

Generally the Support Vector Machine would be the most appropriate model given the small training set size, it generalizes well given the nature of the data, binary and categorical. Training and predicting is very quick so does not consume significant amount of CPU resource, and the model does not consume a large amount of storage resource.

**In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a decision tree or support vector machine, how does it make a prediction).**

From the supplied student dataset, we split it into a training set, containing 75% of the data and a test set with the remaining 25%. Using the training dataset we build the model, this is done using the following steps:

1. In the student data, find the feature that best separates the students into groups with respect to whether they passed or failed. If the feature is binary, ie. the feature can only take on two discrete values then two groups will be created as a result of the split. Typically the number of groups created will be equivalent to the number of values a feature can take on.

2. For each of the groups from step (1): repeat step (1), this time each group is split using one of the remaining features that best separate the students. Steps (1) and (2) are repeated until the groups can nolonger be sub-divided.

The resulting model has a tree like structure where each branch, which were created when a group was split, can be seen as a decision point. So when the model is presented with a student data point, the decision point at the top of the tree will evaluate the appropriate feature value ie. the value of the feature that originally result ed in the decision point's creation. Depending upon the value of the feature the model takes a path to the next decision point or you get the result of the classification for the data point, ie "yes" the student passed or "no" the student failed.

If reaching the next decision point, the value of another feature will be evaluated, again as before this will lead to another decision point or the data point is classified. This process repeats until there are no more decision points, ie. all relevant features in the students data point have been evaluated and the student data point has been classified as either a pass or fail.

The chosen model, the GradientBoostingClassifier, builds many such trees and uses a combination of all their results inorder to build the final model. The resulting model gives more accurate predictions when compared to a model built on a single tree.

Now that the model has been built using the training set, we need to know how it performs on previously

unseen data. To test this we need to apply it to data points that it has never seen before. This previously unseen data is the test data, created earlier as a subset of the original dataset. Ideally the model should perform similarly on both the training and test data. As the test data is a subset of the original dataset, each data point contains the feature “passed”, this can be used to test the models accuracy by comparing it with the predicted value for the feature when the model is given the test data to evaluate.

**Fine-tune the model. Use gridsearch with at least one important parameter tuned and with at least 3 settings. Use the entire training set for this.**

See the python notebook for details.

**What is the model’s final F1 score?**

The final F1 score for the tuned model was 0.822580645161.