# 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?

**Answer**

It is a classification problem because the goal is to classify students in two groups. The ones that need early intervention; and the ones that the early intervention is not needed.

# Exploring the Data

Can you find out the following facts about the dataset?

| **Variable** | **Answer** |
|---|---:|
| Total number of students | 395 |
| # of students who passed | 265 |
| # of students who failed | 130 |
| # of features | 31 |

| | |
|---|---|
| Graduation rate of the class (%) | 67.0 |

# Training and evaluating the models

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

## 1. Gaussian Naive Bayes

1. What are the general applications of this model? It is a classification algorithm that assigns each observation to the most likely class given its predictor values.
2. What are its strengths and weaknesses?
    1. **Strengths:** Easy to train.
    2. **Weaknesses**: Assumption of independence of attributes is constraining. For example, there is a high correlation between the father and the mother’s education. These two attributes are arguably not independent.
3. Given what you know about the data so far, why did you choose this model to apply? It is intuitively easy to explain the results.
4. Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score.

| | **Training set** | **Test set** |
|---|---|---|
| F1 | 79% | 75% |

1. 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.

| **Variable** | **Training set size** | | |
|---|---|---|---|
| | **100** | **200** | **300** |
| Training time (secs) | 0.001 | 0.001 | 0.001 |
| Prediction time (secs) | 0 | 0 | 0 |
| F1 score training set | 81.9% | 80.9% | 79.0% |
| F1 score test set | 78.1% | 75.2% | 71.3% |

## 2. Support Vector Machine

1. What are the general applications of this model? It is an algorithm that performs classification tasks by constructing hyperplanes in a multidimensional space that separates cases of different class labels.

2. What are its strengths and weaknesses?
   1. **Strengths:**
      1. High accuracy
      2. With an appropiate kernel SVM work well even when data is not linearly separable

    2. **Weaknesses**:
        1. Finding the right kernel might be difficult
        2. Training takes longer than Naive Bayes.
3. Given what you know about the data so far, why did you choose this model to apply? It much have higher accuracy than Naive Bayes

4. Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score.

| | **Training set** | **Test set** |
|---|---|---|
| F1 | 79.0% | 77.8% |

1. 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.

| **Variable** | **Training set size** | | |
|---|---|---|---|
| | **100** | **200** | **300** |
| Training time (secs) | 0.001 | 0.003 | 0.006 |
| Prediction time (secs) | 0.001 | 0.002 | 0.007 |
| F1 score training set | 87.3% | 89.2% | 88.3% |
| | | | |

| | | | |
|---|---|---|---|
| F1 score test set | 81.2% | 76.4% | 77.8% |

## 3. Decision Tree

1. What are the general applications of this model? It is algorithm that predict the value of a target variable by learning simple decision rules inferred from the data features.

2. What are its strengths and weaknesses?
    1. **Strengths:**
        1. Perhaps the most easiest model to interpret and explain
        2. It is non-parametric so that the date his non linearly separable is not a concern
    2. **Weaknesses**:
        1. Easy to overfit the data
        2. Training takes longer than Naive Bayes.
3. Given what you know about the data so far, why did you choose this model to apply? It much have higher accuracy than Naive Bayes

4. Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F1 score.

| | **Training set** | **Test set** |
|---|---|---|
| F1 | 79.0% | 68.9% |

1. 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.

| Variable | Training set size | | |
|---|---|---|---|
| | **100** | **200** | **300** |
| Training time (secs) | 0.001 | 0.001 | 0.002 |
| Prediction time (secs) | 0.000 | 0.000 | 0.000 |
| F1 score training set | 100% | 100% | 100% |
| F1 score test set | 72.6% | 69.9% | 68.9% |

# Choosing the Best Model

Based on the experiments you performed earlier, in 2–3 paragraphs explain to the board of supervisors what single model you choose as the best model. Which model has the best test F1 score and time efficiency? Which model is generally the most appropriate based on the available data, limited resources, cost, and performance? Please directly compare and contrast the numerical values recored to make your case.

**Answer**

I would choose the SVM model because it has the best F1 score, and does not take much time to training it. That is, this model classifies correctly

78% of the students based on their information and is fast. Although estimating this model takes longer that the others, in practice, the difference is less than a fraction of a second.

Additionally the prediction gains are very noticeable. The SVM model has an accuracy of 78%, 9 percentage points above the tree model and 3 percentage points above the naive Bayes model.

How does this model work? Support vector machines classifies the students according to their attributes. Suppose we have two attributes and two categories as shown in Figure 1. However, a line that passes too close to any of the points could have difficulties to generalize the classification problem. SVM finds the line that passes as far as possible from all points. Thus, SVM finds the hyperplane (in this example the line) that gives the largest minimum distance to the training examples. Twice this distance is a concept in SVM called *margin*, and SVM finds the hyperplane that maximizes it.

**Figure 1.** Many separation hyperplanes

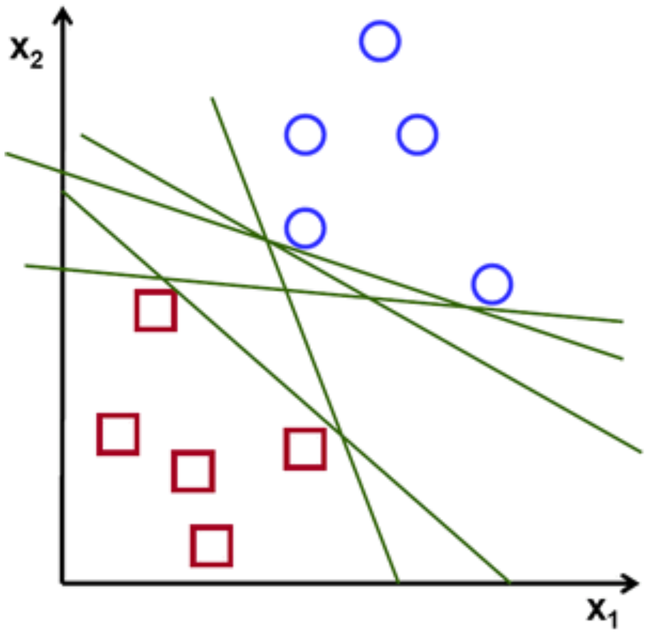
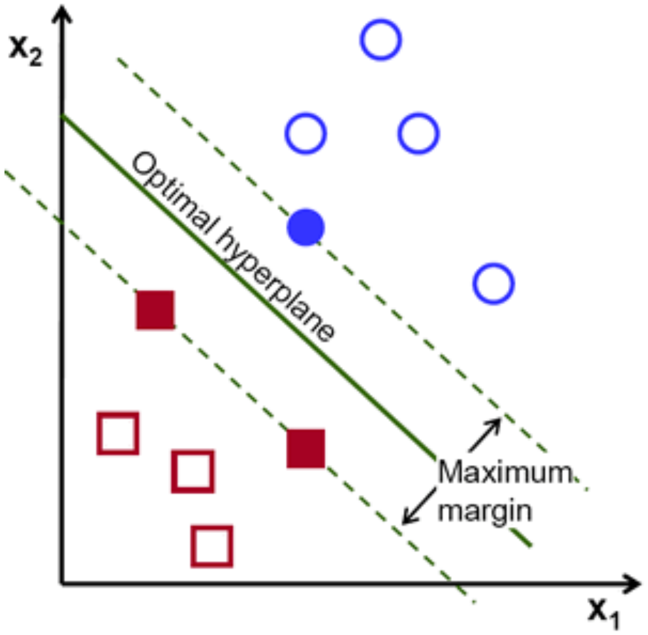
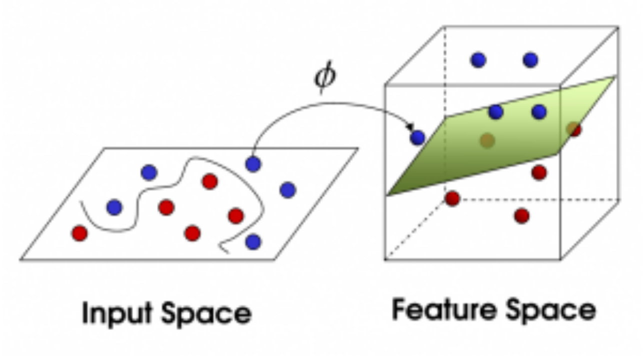


**Figure 2.** SVM finds the hyperplane that maximises the margin



Sometimes, there is not a linar function that separates the two categories. In this case, SVM has a method, called the kernel trick, that projects the input space into a higher dimensional space. This projection is done through a non-linear function $\varphi(x)$ as shown in Figure 3. SVM, in this case, finds the separator hyperplane of all projected data.

**Figure 3.** The kernel trick consists in project the input space in a higher dimensional space so that SVM finds a hyperplane that separates the categories



A hyperplane in a two dimensional space is defined by the equation:

$\beta0 + \beta1X1 + \beta2X2 = 0$

SVM finds the parameters $\beta$. In order to classify an observation one multiply each feature with its corresponding parameter. A test observation is assigned a class depending on whether the result of this operation is positive or negative, that is, on which side of the hyperplane is located.

---

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.

What is the model’s final F1 score?

**Answer**

The F1 score improved from 77.8% to 80.8% after tuning the model. A penalty value of 10, and a gamma of 0.001 were the optimal values in the grid search.

---

Some figures were taken from OpenCV