1. 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?

The algorithm should identify who needs an early intervention and who doesn't. This means that the output of our machine learning algorithm should be 1 for who needs an intervention and 0 for who doesn't, which means it needs to classify every student in two distinct classes. This is a classification problem.

2. 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 (%): 67.09%
- Number of features (excluding the label/target column): 30

Use the code block provided in the template to compute these values.

3. Preparing the Data

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

- Identify feature and target columns

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

- Preprocess feature columns

Processed feature columns (48):-

['school_GP', 'school_MS', 'sex_F', 'sex_M', 'age', 'address_R', 'address_U', 'famsize_GT3', 'famsize_LE3', 'Pstatus_A', 'Pstatus_T', 'Medu', 'Fedu', 'Mjob_at_home', 'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_teacher', 'Fjob_at_home', 'Fjob_health', 'Fjob_other', 'Fjob_services', 'Fjob_teacher', 'reason_course', 'reason_home', 'reason_other', 'reason_reputation', 'guardian_father', 'guardian_mother', 'guardian_other', 'traveltime', 'studytime', 'failures', 'schoolsup', 'famsup', 'paid', 'activities', 'nursery', 'higher', 'internet', 'romantic', 'famrel', 'freetime', 'goout', 'Dalc', 'Walc', 'health', 'absences']

- Split data into training and test sets

Training set: 300 samples

Test set: 95 samples

In this step I was having problems computing the target column with string values, so I decided to preprocess the target column as well.

To split the dataset into training and test set I decided to use the method provided by the library because it maintain the proportion of positive and negative examples in both sets to the same level of the original dataset.

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-leatarter code snippets for these steps have been provided in the template.rn, 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?

- **DecisionTreeClassifier:** Decision Trees are low complexity classifiers capable of model datasets that are linearly separable or non/linearly separable. However, the decision boundary computed by decision trees are always composed by multiple lines, which might work fairly well on some types of datasets and worse on others. This dataset might be one of the cases where decision trees perform better than other algorithms.

- **SVC (Support Vector Classifier)**: Support Vectors Machines provide very high accuracy in classification with fairly low complexity especially in case of binary classification. This is going to be slower than the decision trees but it might provide results that are more accurate than many other models with a fairly acceptable complexity. This parametric classifier tries to find a mathematical function that will represent the boundaries for the classification maximizing the distance between the boundary function and the points of each different class. The complexity of the boundary function, commonly called kernel function, can be selected automatically through the cross validation process according to the dataset we are working on. This enables the model to provide better results of many other classifiers that do not take into consideration the margin, and keeps the complexity of the solution found as low as possible selecting the right kernel.

- **KNeighborsClassifier:** The third classifier I am going to use is a non-parametric classifier. In some cases the classification problem is so complex that is impossible to find a mathematical function that separates the data with a good accuracy. In those cases a lazy learner such as K Neighbors Classifier might achieve a better classification of the points compared to the other models I have used so far. If the previous classifiers don't perform very well this would be the next classifier I would test on the dataset. K-Nearest Neighbors will look for the similar cases and try to make a better prediction based on only those few instances. However, this classifier presents some disadvantages: the training time is almost instantaneous because all the algorithm does is storing the whole dataset in memory. The prediction time higher than the eager learners because the algorithm has to calculate the similarity between the given point and all the other points of the dataset, select the most similar points and calculate the prediction considering only this small subset. This process has to be executed at every prediction. Furthermore, this adds also another problem, which is the choice of the similarity function. Examples might be the Euclidean distance or the Manhattan distance. This choice has an impact on the prediction time, even though it is still fairly small.

•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.

Classifier	train_size	F1_train	F1_test	train_time	pred_time
Decision Tree	300	0.825397	0.764706	0.0015049	0.000114918
Decision Tree	196	0.75	0.764706	0.000935078	0.000123978
Decision Tree	100	0.785185	0.66087	0.000810862	8.10623e-05
SVC	300	0.973494	0.807692	0.0114839	0.00839496
SVC	196	0.989899	0.678899	0.0045588	0.00340009
SVC	100	1	0.797468	0.00143194	0.000952959
Nearest Neighbors	300	0.825263	0.781457	0.000540018	0.00747395
Nearest Neighbors	196	0.753623	0.642857	0.000383854	0.0033021
Nearest Neighbors	100	0.810127	0.781457	0.000307083	0.00127506

5. 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 recorded to make your case.

In 1-3 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 learn to make a prediction).

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?

The first thing I noticed on the results is that decision trees don't perform better with more data: the F1 score on the test set does not change with a dataset of 200 or 300 instances. This means that increasing the data I will not be able to get anything better than the performance I got on the datasets I used so far.

The second thing I found very interesting is that the other two models show a lower F1 score on the 200 instances dataset compared to the 100 and 300 instances datasets. This might mean that the features are not so related to the target or there is too much noise in the dataset.

Support Vector Machine is clearly overfitting the training set on all three training sets. This means that increasing the size of the dataset we can obtain a much better results from this model.

The K Nearest Neighbors is performing similar to the Support Vector Machine. SVC seems fitting better the training set but this result depends on the parameters we used in our classifier. Despite the training time is higher, SVC shows a lower prediction time. Increasing the dataset size, we can expect it to be close to constant, while for K Nearest Neighbors it will increase in function of the size. I think that higher training time is more acceptable than higher prediction time because the training has to be performed just once.

We should consider that a simple learner f(x)=1 would achieve an F1 score of 0.8030303, which means we should expect a score higher than 0.8 from a good model.

The Support Vector Machine seems to be the best choice for this classification problem.

The final F1 score is:

F1 score for training set: 0.838427947598

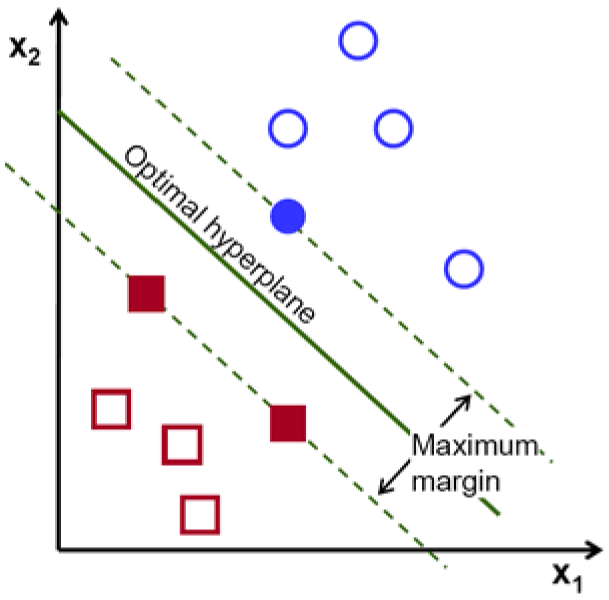
F1 score for test set: 0.813333333333

Support Vector Classifier

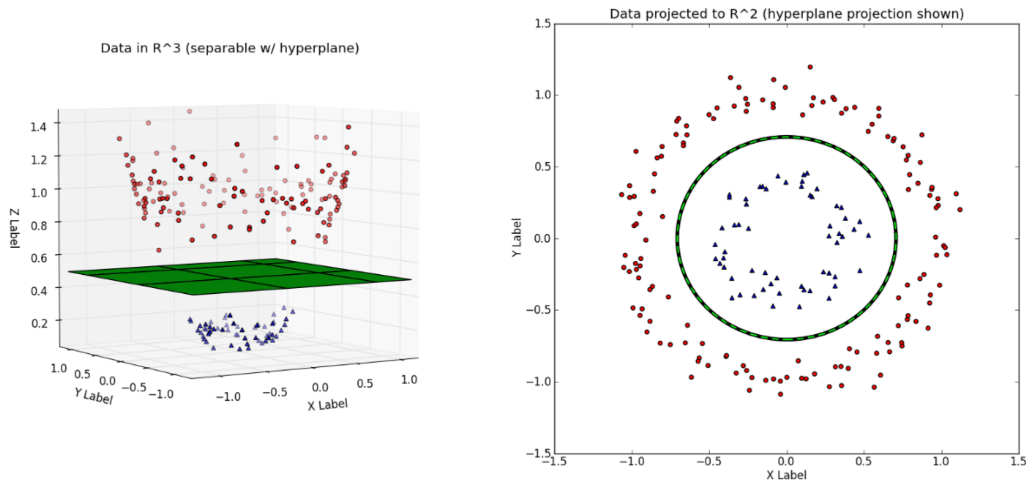
The SVC tries to divide the dataset into two or multiple classes finding the function that better divides the two sets, which means it has a larger margin between the points of the classes.

Let's explain how this classifier works with an example. Let's say we are trying to make a learning machine that predicts when a certain event is going to occur. This event Y depends only on the value of one variable X and our dataset is composed by only two instances: (X=50, Y=0), (X=100, Y=1). If we have to calculate a decision boundary based on these two instances only, we would probably choose to divide the classes for {Y = 1 | X > 75} because it is a better generalization compared to {Y = 1 | X > 99} and, even though they perform exactly in the same way onto the training set, it will probably perform better onto the test set and new data.

If we generalize this concept on multi-dimensional space we obtain that SVC will calculate the optimal hyperplane that divides the data maximizing the margin between the boundary function and the two classes. Here is a bi-dimensional example:



When the data are not linearly separable we can introduce new features that are polinomially generated from the initial ones. This means that if my dataset composed by X1 and X2 and it is not linearly separable, I can generate features such as X3=X1^2, X4=X2^2, X5=X1*X2 and use X1 to X5 as features. The new dataset will be linearly separable and the function calculated will correspond to a non-linear boundary function. A graphical example of a non-linear bi-dimensional problem is reported below:



However, computing the additional features might be very computationally expensive. SVC allows the programmer to change the kernel function with a non-linear function. In this way, the input features are still the same but the kernel function will be looking for a non-linear boundary. The kernel function can be also automatically selected through cross validation.

When we ask a trained SVM machine to give a prediction on a point, the machine will consider the position of the point from the boundary. The label assigned to the instance will be depending on the position of the point in relation with the boundary function.