P2: Build A Student Intervention System

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?

In the input student-data.csv file, the last column "passed" is designated as the target/label and consists of only discrete values (i.e. "yes" and "no"). Also, the objective seems to involve classifying the observations with the aforementioned discrete values. Therefore, this supervised machine learning problem is classification.

On the other hand, regression problems are ideal for handling target/label consisting of continuous values.

2. Exploring the Data

Can you find out the following facts about the dataset?

- *Total number of students*
- *Number of students who passed*
- *Number of students who failed*
- *Graduation rate of the class (%)*
- *Number of features (excluding the label/target column)*

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%

3. 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*

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, 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?*
- *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.*

***Note**: You need to produce 3 such tables - one for each model.*

Model 1: K-NN Classifier			
Strengths	• No learning required since the training data itself is used for classification at query time (lazy learning) • Easy integration of additional training data		

Weaknesses	• Slow prediction due to needing to compute distance vs. training data for each query
	  ○ O(n) + k running time per query (if unsorted data points)
	  ○ O(log n) + k running time per query (if sorted data points)
	• Requires storage for training data
	  ○ O(dn) space (where d := dimension)
	  ○ As number of features or dimensions grows, the amount of data needed to generalize accurately grows exponentially 2^d (curse of dimensionality)		

Reason For Choosing This Model	• Simplest machine learning algorithm as it involves assigning the most frequent label among k training samples nearest to a query point		
Performance			
	Training Set Size		
	100	**200**	**300**
Training Time (secs)	0.002	0.001	0.001
Prediction Time (secs)	0.002	0.003	0.005
F1 Score for Training Set	1.0	1.0	1.0
F1 Score for Test Set	0.819444444444	0.828571428571	0.820143884892

Model 2: Logistic Regression			
Strengths	• Very fast prediction ○ O(1) running time • Low risk of overfitting		
Weaknesses	• Assumption of linear decision boundary		
Reason For Choosing This Model	• Known to be a fast classifier at query time		
Performance			
	Training Set Size		
	100	**200**	**300**
Training Time (secs)	0.004	0.003	0.004
Prediction Time (secs)	0.000	0.000	0.000
F1 Score for Training Set	0.850393700787	0.785123966942	0.794736842105
F1 Score for Test Set	0.738461538462	0.77519379845	0.755905511811

Model 3: Support Vector Machine (SVM)			
Strengths	• Decision boundary can be linear or non-linear via kernel support, resulting in better model fitting on training data • Effective in high dimensional spaces • Memory efficient; depends on only a subset of the training data		

Weaknesses	• Compute and storage requirements increase rapidly with the number of training data
	  ○ Quadratic programming problem
	  ○ Scales between O(dn^2) and O(dn^3), where d := dimension
	• Risk of overfitting if parameters are not calibrated properly		

Reason For Choosing This Model	• Known to be a powerful classifier where, using the proper kernel function, it can result in better model fitting on training data		
Performance			
	Training Set Size		
	100	**200**	**300**
Training Time (secs)	0.004	0.007	0.042
Prediction Time (secs)	0.001	0.008	0.003
F1 Score for Training Set	0.859504132231	0.857142857143	0.86
F1 Score for Test Set	0.7	0.790697674419	0.814814814815

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.

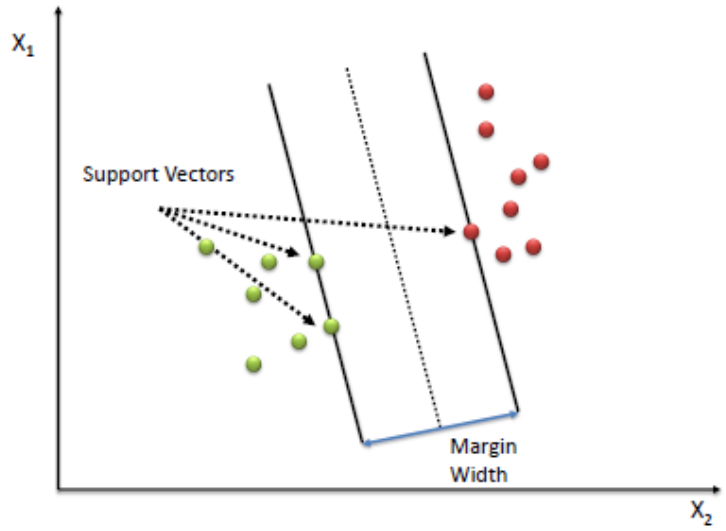
What is the model's final F1 score?

K-Nearest Neighbor, Logistic Regression, and Support Vector Machine classification algorithms were evaluated on the provided input data. Based on the experiments, K-Nearest Neighbor classification reported the best test F1 score (0.820143884892 on training set size 300), followed by Support Vector Machine classification (0.814814814815 on training set size 300). On the other hand, Logistic Regression had the fastest prediction time (0.000 seconds).

Based on the observations made from the aforementioned experiments, the Support Vector Machine model should be used. Even though its training time was the slowest among the three, the frequency where the training data set is used to fit the classification model is most likely less than amount of times querying the test data set. Furthermore, SVM's the prediction time was the second fastest with 0.003 seconds. Performance-wise, its test F1 score was 0.005329070077 lower than the highest reported test F1 score, i.e.

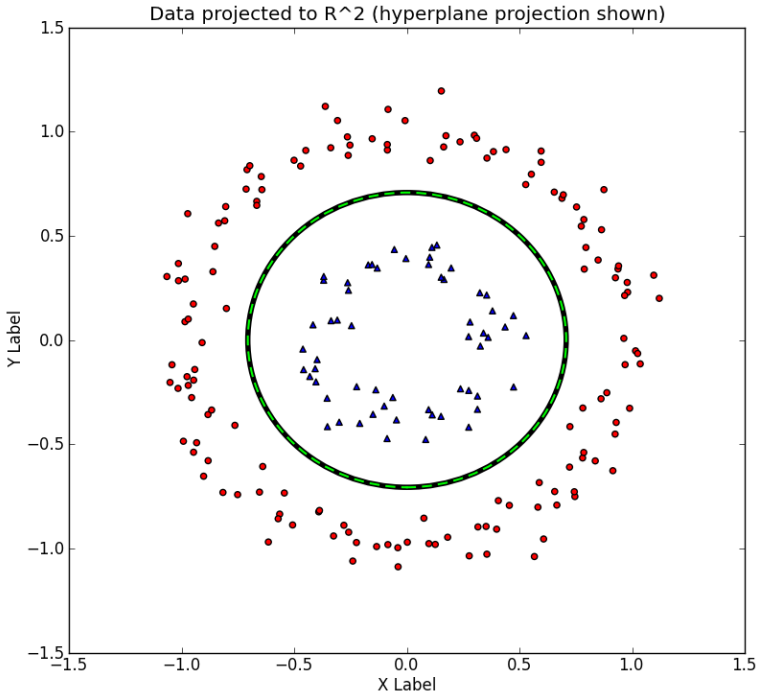
0.814814814815 (SVM) vs. 0.820143884892 (K-Nearest Neighbor) vs. 0.755905511811 (Logistic Regression). Also, Support Vector Machines only depends on a subset of the training data, therefore being a memory efficient.

The Support Vector Machine (SVM) can be defined as an algorithm that classifies new observations based on the labelled training data points. To better understand this algorithm, let's consider the following graph that illustrates a simple Support Vector Machine:

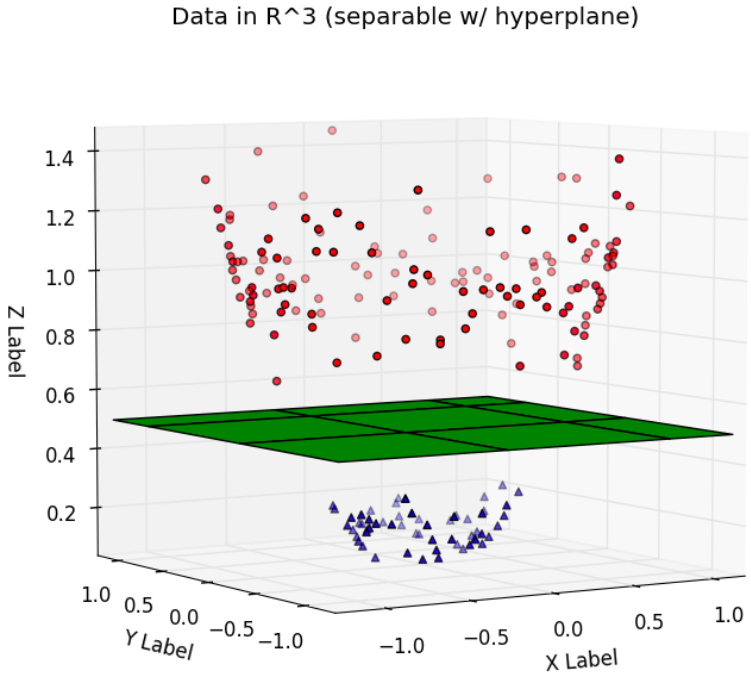


The circles represent the data points classified as either green or red from the training data set and are divided by a straight dotted line based on their colour. Although there are an infinite number of possible lines that can accomplish this task, the Support Vector Machine, in particular, finds one with the maximum margin width defined by the training data points closest to it (support vectors). The label of a test data point can be predicted by determining which side of this dividing line it lies on and take the associated class.

Unfortunately, it is not always possible to find a straight line in 2-dimesions that can divide the training data set based on its labels, as illustrated below:



At 2-dimensions, the classified training data points are divided by a non-linear separator (a circle in this case). To remedy this, the Support Vector Machine can project the training data points into higher dimensions using the kernel trick and, as demonstrated in the 3-dimension graph below, find a kind of higher-dimensional “line” (i.e. plane in 3-dimensions, “hyperplane” in higher dimensions) with the maximum margin width that can “linearly” divide the labelled training data points at a higher dimension:



Final F1 Score: 0.837837837838