P2: BUILD A STUDENT INTERVENTION SYSTEM

Question 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?

ANSWER: We are dealing with classification as supervised machine learning problem because we are taking some inputs and we are mapping them to a discrete variable, passed, which can take 2 values: passed or failed.

Question 2. Exploring the Data

Now, 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

ANSWER:

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%

A very peculiar characteristic of this dataset is that its labels are quite unbalanced. We are provided with data about 395 students, 265 of whom went on to pass while 130 ended up failing. We then have a graduation rate of 67.09%. This means that the number of positive training examples is much more than the number of negative training examples or the number of students who passed is much more than the number of students who failed. This is the main reason why the accuracy metric might not be ideal for scoring our model. Here, the F1-score will be chosen as the default evaluation metric.

Question 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 F_1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.

Produce a table showing training time, prediction time, F_1 score on training set and F_1 score on test set, for each training set size.

ANSWER:

Model 1: Support Vector Machine – SVM

- **What are the general applications of this model?**
  - Support vector machine is a learning machine with excellent generalization in many learning problems such as in handwritten digit recognition, face recognition, novelty detection, etc...

- **The advantages of support vector machines are:**
  - Effective in high dimensional spaces.
  - SVM's work great when the data is separable using a hyper plane
  - SVM have a regularisation parameter which makes think about avoiding overfitting
  - SVM is Still effective in cases where number of dimensions is greater than the number of samples.
  - SVM have kernel trick enabling to build in expert knowledge of a problem via engineering the kernel, modelling non-linear relationship for example.
  - Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
  - Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

- **The disadvantages of support vector machines include:**
  - If the number of features is much greater than the number of samples, the method is likely to give poor performances.
  - Need a good kernel function
  - They are sensitive to noise; a relatively small number of mislabeled examples can dramatically decrease the performance
  - They consider only two classes.
  - SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

- **Given what you know about the data so far, why did you choose this model to apply?**
    - We expected the data to not be linear, given that there are so many features. Using an appropriate kernel like rbf, we would be able to effectively tune the classifier for high f1 score.
      I specifically chose SVM to attack the problem of unbalanced data because SVM is based on strong theoretical foundations and it performs well with moderately unbalanced data even without any modifications. Its unique learning mechanism makes it an interesting candidate for dealing with unbalanced datasets, since SVM only takes into account those instances that are close to the boundary, i.e. the support vectors, for building its model. This means that SVM is unaffected by non-noisy negative instances far away from the boundary even if they are huge in number.

- **Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F_1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.**

SVC	Training Set Size		
	100	200	300
Training Time (secs)	0.001	0.004	0.008
Prediction Time (secs)	0.001	0.001	0.002
F1 Score for training set	0.89932885906040272	0.88599348534201972	0.8717948717948717
F1 Score for testing set	0.80000000000000004	0.79452054794520555	0.78378378378378377

As the training size increased, training and prediction times also decrease intuitively. The F1-scores, both from training and testing decrease slightly as the training set size increase.

Model 2. Decision Trees

- **What are the general applications of this model?**
    - Decision trees are commonly used in operation research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning. Decision Trees are heavily used in credit scoring, in the assessment of probability of default, the loss given default as well as the exposure at default.

- **The advantages of Decision Trees are:**

  - Decision Trees easily handle irrelevant attributes through information gain which is a commonly used metric for decision tree learning.
  - Decision Trees are robust against skewed distributions, since they do not make assumption on the variable's distribution when constructing axis splits.
  - Decision trees rare flexible in handling items with a mixture of real-valued and categorical features, and items with some missing features.
  - Decision Trees focus on the relationship among various events and thereby, replicate the natural course of events, and as such, remain robust with little scope for errors, provided the inputted data is correct.
  - Decision trees also allow for classification of data without computation, can handle both continuous and categorical variables, and provide a clear indication of the most important fields for prediction or classification.
  - Decision trees also allow for partitioning data in a much deeper level, not as easily achieved with other decision-making classifiers such as logistic regression or support of vector machines.
  - Decision trees make explicit all possible alternatives and trace each alternative to its conclusion in a single view, allowing for easy comparison among the various alternatives. The use of separate nodes to denote user defined decisions, uncertainties, and end of process lends further clarity and transparency to the decision-making process.

- **Weaknesses**
  - Decision Trees do not work well if we have smooth boundaries that mean they work best when we have discontinuous piece wise constant model. If we truly have a linear target function decision trees are not the best.
  - Decision Tree's don't work best if we have a lot of un-correlated variables. Decision tree's work by finding the interactions between variables. If we have a situation where there are no interactions between variables linear approaches might be the best.
  - Data fragmentation: each split in a tree leads to a reduced dataset under consideration. And, hence the model created at the split will potentially introduce bias.
  - High variance and unstable: as a result of the greedy strategy applied by decision tree's variance in finding the right starting point of the tree can greatly impact the final result, that means small changes early on can have big impacts later. So- if for example we draw two different samples from our universe, the starting points for both the samples could be very different (and may even be different variables); this can lead to totally different results.

- **Given what you know about the data so far, why did you choose this model to apply?**
  - The data has a lot of binary features. Decision Trees would fit and create subclasses efficiently.

- **Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F_1 score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.**

Decision Tree	Training Set Size		
	100	200	300
Training Time (secs)	0.001	0.001	0.000
Prediction Time (secs)	0.001	0.000	0.000
F1 Score for training set	1.0	1.0	1.0
F1 Score for testing set	0.6885245901639343	0.6885245901639343	0.655999999999999992

Model 3. Gaussian Naive Bayes

- **What are the general applications of this model?**
  - In spite of their apparently over-simplified assumptions, naive Bayes classifiers have worked quite well in many real-world situations, famously document classification and spam filtering. They require a small amount of training data to estimate the necessary parameters.
- **Gaussian Naïve Bayes strengths'.**
  - Gaussian Naive Bayes learners and classifiers can be extremely fast compared to more sophisticated methods. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one dimensional distribution. This in turn helps to alleviate problems stemming from the curse of dimensionality. They are not sensitive to irrelevant features, can handle real and discrete data as well as streaming data well.
- **Gaussian Naïve Bayes Weaknesses':**
  - Some of the Gaussian Naive Bayes Weaknesses include its poor performance if independence assumptions do not hold and has difficulty with zero-frequency values.
- **Given what you know about the data so far, why did you choose this model to apply?**
  - Gaussian Naive Bayes was chosen for its strengths to classify data quickly, which is desired in this scenario to save on computation time.
- **Fit this model to the training data, try to predict labels (for both training and test sets), and measure the F$_1$ score. Repeat this process with different training set sizes (100, 200, 300), keeping test set constant.**

Gaussian Naïve Bayes	Training Set Size		
	100	200	300
Training Time (secs)	0.001	0.000	0.001
Prediction Time (secs)	0.001	0.000	0.000
F1 Score for training set	0.81159420289855067	0.8044280442804429	0.76923076923076916
F1 Score for testing set	0.74380165289256195	0.71666666666666667	0.71074380165289253

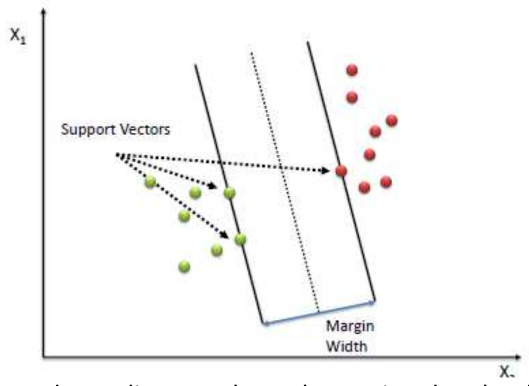
Question5. Choosing the Best Model

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

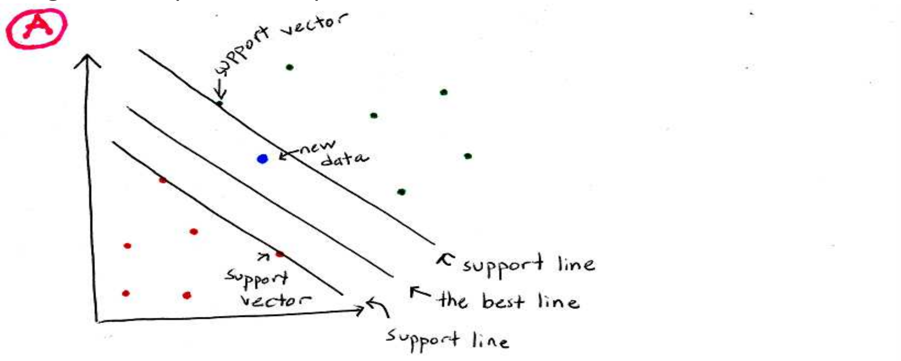
ANSWER:

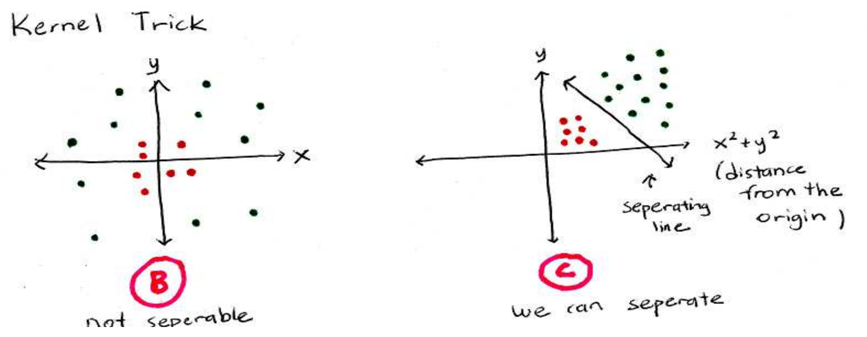
- The best performing model out of these three, on average, in terms of *F*1 score is Support Vector Machine (80.0%). It is better than Decision Tree (68.85%) and Gaussian Naïve Bayes (74.38*%).* Scalability is the capability of an algorithm in handling a growing amount of training data, efficiently. Standard SVM training has O(m^3) time and O(m^2) space complexities, where m is the training set size. It is thus computationally infeasible on very large data sets. By observing SVM in practice, its implementation only approximates the optimal solution by an iterative strategy. If we had to classify thousands or hundreds of thousands of students, training time and prediction time will certainly increase according to the SVM table and the performance will deeply decrease. In order to use *light* SVM, we need to use techniques that reduce the time and space complexities.
  I would argue that the SVM classifier is a better model for our data because it gives a good f1 score get trained fast and also predicts data in negligible time.

- **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).**
  - In layman’s terms, what we want to do with SVM is to find some line that splits the linearly separable data set between the two differently classified groups of data as well as we can. If we think of the greys lines in the figure below, the lines that gets as close as the red and the green lines, what we really need is that the distance between green’s lines is as big as possible. The line in the middle provides the best commitment in the sense that it leaves as much space as possible from the boundaries. Once we get the line that is our classifier.

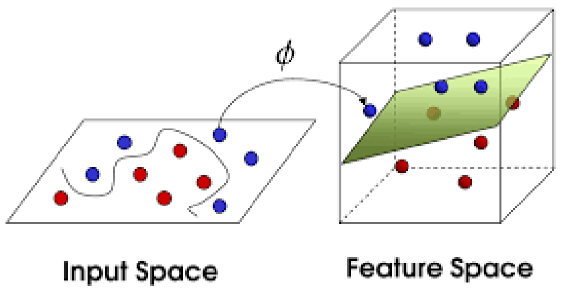


Then depending on where the testing data lands on either side of the line, that's what class we can classify the new data as (see graphic A below) . Then when we get a new point on the plot, we can see which side of the line it's on and decide if it should be on one side or the other. The line that separates them can be straight or curvy. If it's curvy, then we need to use a kernel.





When our data cannot be separable by a straight line(like in the left figure below or the figure B above), SVM takes that data and transform them into higher dimension and then finding a kind of straight line to separate the data in that high space.



In technical terms, this is the kernel trick: it takes the data we give it and transforms it. In goes some great features which we think are going to make a great classifier, and outcomes some data that we don't recognize anymore. This straight line looks like a curvy line when we bring it down to the lower dimensional space where our original data lives in.

- **What is the model's final F_1 score?**
  - The Final F1-score of my model is 0.78912

Bibliography:

https://en.wikipedia.org/wiki/Naive_Bayes_classifier,
http://www.support-vector-machines.org/
https://en.wikipedia.org/wiki/Decision_tree_learning
https://en.wikipedia.org/wiki/Classification_Tree_Method
http://scikit-learn.org/stable/modules/svm.html
https://www.quora.com/What-are-the-advantages-of-using-a-decision-tree-for-classification
https://github.com/rahulravindran0108/student-intervention-system

An Introduction to Statistical Learning with Applications in R, 2014, Gareth James, Daniela Witten Trevor Hastie and Robert Tibshirani.

Modelling Techniques in Predictive Analytics: Business Problems and Solutions with R,2013, Thomas W. Miller.