Machine Learning Engineer Nanodegree

Project 2: Build a Student Intervention System

1. Classification vs Regression

The student intervention system is a classification problem since the output is discrete and is of type pass/fail, as opposed to a continuous range.

2. Exploring the Data

Here are some high-level stats about the data:

- 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

3. Preparing the Data

Please refer to the accompanying ipython notebook for this section's deliverables.

4. Training and Evaluating Models

I chose to train and evaluate the following three classifiers for this problem:

1. Nearest Neighbors (KNN)
2. Support Vector Machine
3. Decision Tree Classifier

Below I will share the findings for each of these classifiers.

i) Nearest Neighbors (KNN)

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

k-NN is an instance-based (non-parametric) lazy learner, where the function is only approximated locally and all computation is deferred until classification. It is also among the simplest ML algorithms and does not make any assumptions on the underlying data distribution which makes it a good first candidate for modeling this problem.

Strengths:

1. lazy learner, fast and minimal training phase
2. it does not make any assumptions on the underlying data distribution

Weaknesses:

1. Costly testing phase in both time and memory.

2. Lack of generalization means that KNN keeps all the training data, which is needed in the testing phase

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

Mainly because KNN is versatile and I don't know much about the underlying distribution and structure of the data at this point.

Q: 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	F1 Score on Training Set	F1 Score on Test Set
KNN with 100 input data	0.8187	0.7625
KNN with 200 input data	0.8742	0.7352
KNN with 300 input data	0.8703	0.7703

ii) Support Vector Machine

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

SVM is a non-probabilistic parametric classifier with a broad range of applications. It uses a linear hyperplane for separating the data points, which can also be used as a non-linear classifier through the use of kernels.

Strengths:

1. It uses the kernel trick, so you can build in expert knowledge about the problem via engineering the kernel
2. SVM is defined by a convex optimisation problem (no local minima) for which there are efficient methods

Weaknesses:

1. Expensive training and testing phase, both in speed and size

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

SVM is a powerful and flexible classifier that could be adjusted to specific applications through fine tuning its parameters. As such, I thought it would be another good candidate for modeling the student intervention system.

Q: 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	F1 Score on Training Set	F1 Score on Test Set
SVM with 100 input data	0.8758	0.7432
SVM with 200 input data	0.8700	0.7482
SVM with 300 input data	0.8690	0.7801

iii) Decision Tree Classifier

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

Decision Tree is also a simple yet powerful classifier which is also easy to interpret and explain

Strengths:

1. Simple and fast learning and prediction phase
2. Decision trees implicitly perform variable screening or feature selection

Weaknesses:

1. May overfit data

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

Given the CPU/memory constraints and given that a simpler model that is easier to interpret and explain to the executives, I thought Decision Tree classifier to be a good potential candidate for this model.

Q: 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	F1 Score on Training Set	F1 Score on Test Set
DT with 100 input data	1.0000	0.6222
DT with 200 input data	1.0000	0.7272
DT with 300 input data	1.0000	0.7540

5. Choosing the Best Model

Q: 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.

Your model will be evaluated on three factors:

- **Its F1 score, summarizing the number of correct positives and correct negatives out of all possible cases. In other words, how well does the model differentiate likely passes from failures?**
- **The size of the training set, preferring smaller training sets over larger ones. That is, how much data does the model need to make a reasonable prediction?**
- **The computation resources to make a reliable prediction. How much time and memory is required to correctly identify students that need intervention?**

The data corresponding to the run times of the learning phase and testing phase and the F1 scores for the train and test data has been captured in the tables below:

Classifier	Training Time	Test Time
KNN with 100 input data	0.002	0.003
KNN with 200 input data	0.001	0.003
KNN with 300 input data	0.002	0.004
SVM with 100 input data	0.005	0.002
SVM with 200 input data	0.005	0.002
SVM with 300 input data	0.010	0.002
DT with 100 input data	0.003	0.000
DT with 200 input data	0.002	0.000
DT with 300 input data	0.003	0.000

Classifier	F1 Score on Training Set	F1 Score on Test Set
KNN with 100 input data	0.7517	0.7625
KNN with 200 input data	0.8210	0.7352
KNN with 300 input data	0.8610	0.7703
SVM with 100 input data	0.8211	0.7432
SVM with 200 input data	0.8441	0.7482
SVM with 300 input data	0.8620	0.7801
DT with 100 input data	1.0000	0.6222
DT with 200 input data	1.0000	0.7272
DT with 300 input data	1.0000	0.7540

The following insights could be drawn from the data:

1. kNN has fast learning phase, but a relatively expensive prediction phase mainly because it's an instance based learner.
2. SVM run times for both the learning and testing phase seem to grow proportional to the square of the input size, which is more sensitive to the input size compared to the other two models.
3. DT has really fast learning and testing phase, however it suffers from overfitting and has low F1 score on the test data.

My suggestion is to pick the SVM classifier for the following reasons:

1. This classifier performs relatively well for smaller training sets as hinted by the F1 test score

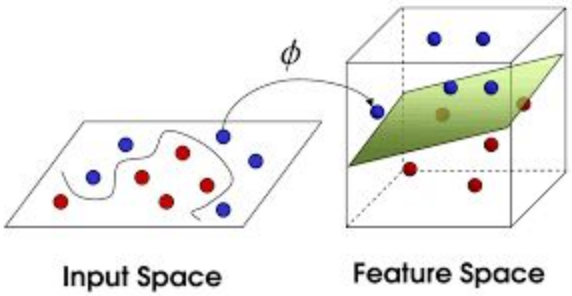
2. The fact that the training/learning run times for SVM are comparable to kNN and DT and in the same order of magnitude, with the advantage of using less memory due to its being a parametric learner.

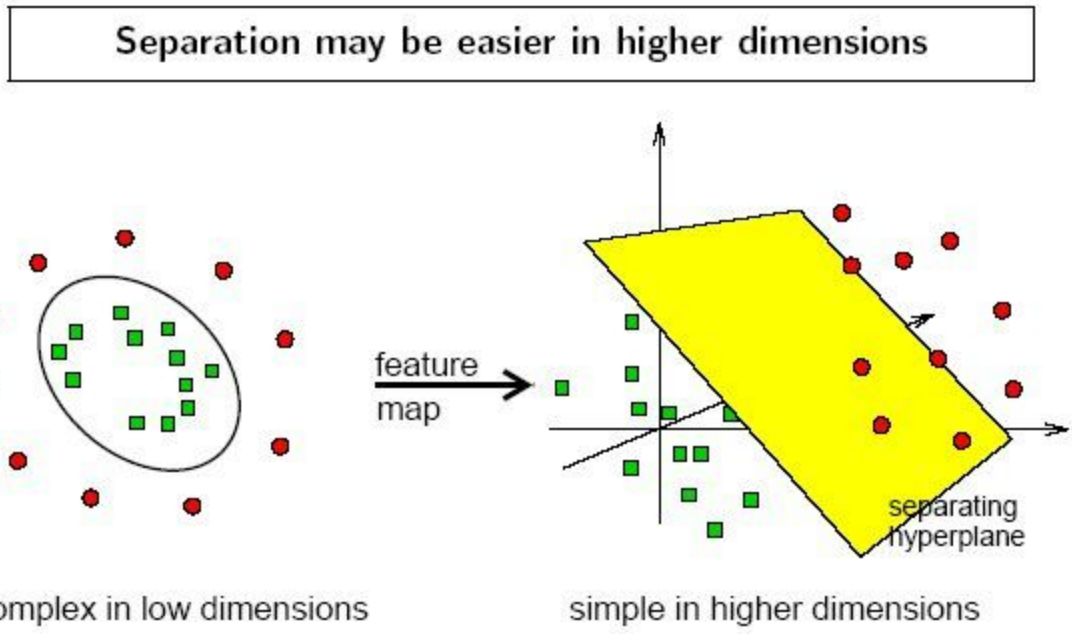
Q: In 1-3 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a decision tree or support vector machine, how does it learn to make a prediction).

The support vector machine model used for the student intervention system works in the following way:

1. Assuming data samples are floating balls in a room, the model learns to find a separating plane that best separates white balls (passing students) from black balls (non-passing students).
2. When trying to predict, the model simply determines which side of the plan the new ball falls into, and classifies it as similar to category of the balls in that side.

There might be situations where it might not be possible to find a line that clearly separates the data on a plane in case of 2D input space (or a plane that separates data in a 3D input space). In such cases, a trick called Kernel can be used to project inputs into higher space by adding new features to the input data, so the linear separation of data is possible or easier. For example in the figures shown below, the original data lies on a plane (2D) and not separable with a straight line, but when projected to a 3D space using a Kernel, the data could be separated by a plane.





Q: 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?

Prediction time (secs): 0.003
F1 score on test set for this CLF is: 0.**738255033557**

Fine tuned CLF is : GridSearchCV(cv=None, error_score='raise',
estimator=SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False),
fit_params={}, iid=True, n_jobs=1,
param_grid={'kernel': ('linear', 'rbf'), 'C': [1, 10, 100], 'gamma': [0.001, 0.01]},
pre_dispatch='2*n_jobs', refit=True,
scoring=make_scorer(f1_score, pos_label=yes), verbose=0)
{'n_jobs': 1, 'verbose': 0, 'estimator__gamma': 'auto', 'estimator__decision_function_shape': None, 'estimator__probability': False, 'param_grid': {'kernel': ('linear', 'rbf'), 'C': [1, 10, 100], 'gamma': [0.001, 0.01]}, 'cv': None, 'scoring': make_scorer(f1_score, pos_label=yes), 'estimator__cache_size': 200, 'estimator__verbose': False, 'pre_dispatch': '2*n_jobs', 'estimator__kernel': 'rbf', 'fit_params': {}, 'estimator__max_iter': -1, 'refit': True, 'iid': True, 'estimator__shrinking': True, 'estimator__degree': 3, 'estimator__class_weight': None, 'estimator__C': 1.0, 'estimator__random_state': None, 'estimator': SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
decision_function_shape=None, degree=3, gamma='auto', kernel='rbf',
max_iter=-1, probability=False, random_state=None, shrinking=True,
tol=0.001, verbose=False), 'estimator__coef0': 0.0, 'error_score': 'raise', 'estimator__tol': 0.001}