PROJECT 2: SUPERVISED LEARNING

BUILDING A STUDENT INTERVENTION SYSTEM

1. CLASSIFICATION VS REGRESSION

This is a **classification** problem. We need to identify whether a student will most likely pass or fail. The model should therefore output a discreet labeled value being either passed or not.

2. EXPLORING THE DATA

The dataset has the following facts:

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

The dataset has a total of 31 columns. The last column "passed" is the target column and the other 30 are the features that we are going to use to predict the target label.

Some of the feature columns have non-numeric categorical values. However, our machine learning tool is expecting numeric values. These columns are converted to have Bernoulli distribution; i.e. one column for each category with values 1 if the value falls on the same category, 0 otherwise. After converting, the dataset now has a total of 48 features.

The dataset is then randomly split into training and testing subsets. 300 of the data points are allocated for training, and 95 for testing.

4. TRAINING AND EVALUATION MODELS

Decision Tree Classifier

A decision tree classifier is a recursive, partition-based tree model that predicts the target output by learning from the features of the dataset through simple rules. It works best for dataset that can be easily split based on their features.

Decision trees are simple to understand and interpret, and the tree model that it learned can be visualized. Prediction time depends on the number of data points it was trained on. The data points are recursively split to form the tree, thereby making prediction time logarithmic in the number of data points.

Decision trees can easily overfit. Small variation of features in the dataset can generate a very different tree model.

Decision tree classifier might be good to apply for this data set given that the simplistic values of the features which we can easily follow to make sense of the predicted output. Also, with just 300 training data points, predicting with decision tree is relatively fast.

	Training set size		
	100	200	300
Training time (sec)	0.001	0.003	0.002
Prediction time (sec)	0.0	0.0	0.0
F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.667	0.624	0.661

Support Vector Machine Classifier

Support vector machine finds the hyperplane that separates the classes with biggest margin possible. SVM is able to separate classes which are not linearly separable through the use of kernel trick. Training time can be computationally intensive. Only a subset of the dataset (support vectors) are kept which are to be used for making decision, hence space requirement is still minimal, and also determines the prediction time for new data points. The complex transformation of data makes SVM great, however these transformation and the resulting boundary plane also often hard to interpret.

SVM is good for our dataset since our data points and number of features is not huge, so we don't have to worry with computation cost very much while we can take advantage of the advance capability of SVM in finding non-linear relationship of our dataset.

	Training set size		
	100	200	300
Training time (sec)	0.002	0.007	0.008
Prediction time (sec)	0.001	0.002	0.002
F1 score for training set	0.863	0.862	0.88
F1 score for test set	0.797	0.792	0.803

K-Nearest Neighbors Classifier

K-Nearest neighbors (KNN) predicts the label based on the labels of its closest neighbors with some notion of distance. It is often very useful when the decision boundary is very irregular. It can take any dataset that doesn't seem to have a general pattern. Training time for KNN is fast, but requires significant amount of space. It basically just remembers all the data points; stores them without any calculations. If there is a large data points, KNN can have slow prediction time. KNN also can't identify which features are more important.

KNN can be good for our dataset since we don't have a huge number of data points in our training set, so the slow prediction time of KNN may not have a big impact. Students with similar features are most likely fall on the same class, and KNN is good at identifying these.

	Training set size		
	100	200	300
Training time (sec)	0.001	0.001	0.001
Prediction time (sec)	0.001	0.002	0.004
F1 score for training set	0.835	0.84	0.863
F1 score for test set	0.752	0.764	0.764

5. Choosing the Best Model

Among the 3 models, SVM produced the best F_1 score of 0.803. The decision tree has perfect F_1 score on training set but have poor performance on testing set. There is a clear evidence of overfitting. KNN shows a rather stable performance even at low number of data points; the F_1 scores (≈ 0.76) have closely similar values across different training set size. SVM also have somewhat consistent F_1 scores (≈ 0.79) across different training set size, but yields better values compared to KNN.

Decision trees runs the fastest in terms of prediction time, but based on it's F_1 scores, it is unreliable. SVM trains much longer than KNN, but SVM predicts faster than KNN. In using the model, training will be done just once, while prediction will be done repeatedly, so it makes sense to give more weight on prediction time than on training time.

Given what we have on our models' performance, time efficiency on prediction, and space requirements, it is best to choose **SVM** classifier as our final model.

SVM finds the largest gap between classes, known as the margin. Basically we put a line in the middle of the gap between the classes. This line effectively separates the classes. The points that are closest the separating line are called support vectors. These support vectors are used to predict the future data points as to which class they would belong. The class of a new data point is determined by which side of the line it fell.

In many cases, our data cannot be separated by a straight line. One powerful feature of SVM is *kernel trick*. A kernel trick is function that transforms our data into a higher dimension so that it will be linearly separable. When projected back to original dimension, we can see that the separating line is not a straight, rather it curves it's way around the data points to separate the classes. A kernel function is just any function that returns a number. So we can replace our kernel with any function that returns some notion of similarity between points. This makes SVM flexible and powerful.

After fine-tuning our model, the final F_1 score is 0.8108.