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?

- This is a classification problem as there is a discrete number of outcomes/labels rather an a continuous value.
- In this case it is binary classification problem as there are 2 possible outputs labels, pass and fail

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**

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

```
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**

Starter code snippets for these steps have been provided in the template.

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 Nearest Neighbours (KNN)

KNN can be used for regression and classification. In this project I am using it for classification. I chose to use this classifier as it is simple, well understood and fast.

KNN takes an average of its neighbours values to make a prediction. This approach automatically smooths out values, which generalises more with an increasing N.

Pros

- Training time is extremely low for any number of samples

Cons

- Prediction time increases in proportion to with N and the number of samples
- Suffers curse of dimensionality
- Works poorly with a large number of features
- Space requirements are exponential with regards to the number of features

	N Samples	Train Time	Pred Time Train	F1 Train	Pred Time Test	F1 Test
KNeighborsClassifier	100	0.0009	0.0020	0.8154	0.0017	0.7519
	200	0.0009	0.0038	0.8252	0.0020	0.7724
	300	0.0009	0.0071	0.8519	0.0026	0.7801

Model 2 - Support Vector Machines (SVM)

An SVM model is used for classification. I chose it as it works well out of the box and also works very well for a large number of features.

SVM's use non linear decision boundaries to split data and will find optimum boundaries by maximising the space between positive and negative samples.

Pros

- Works well with a large number of features
- Can be highly accurate out of the box

Cons

- Training time is very slow and increases exponentially for increasing samples

	N Samples	Train Time	Pred Time Train	F1 Train	Pred Time Test	F1 Test
SVC	100	0.0018	0.0012	0.9008	0.0011	0.8235
	200	0.0042	0.0031	0.8542	0.0016	0.8408
	300	0.0080	0.0061	0.8590	0.0020	0.8408

Model 3 - Decision Tree

A decision tree can be used for classification and regression. In this project I will be using it for classification. I chose it as it is easy to understand, fast and is suited for binary classification problems.

Pros

- Fast to train

- Extremely fast to predict
- Does not suffer the curse of dimensionality

Cons

- Will overfit to the training set
- Does not work well with out of the box parameters

	N Samples	Train Time	Pred Time Train	F1 Train	Pred Time Test	F1 Test
DecisionTreeClassifier	100	0.0009	0.0003	1	0.0003	0.7484
	200	0.0016	0.0004	1	0.0003	0.7038
	300	0.0022	0.0005	1	0.0002	0.7311

5. 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?

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).

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?

Recommendation

After reviewing 3 different classifiers, KNN, Decision Tree and SVM with the student data I have taken into account training time, prediction accuracy and space required.

The KNN model worked well with the f1 test score of 0.7801 when using a sample size of 300. This places it in the middle with the Decision Tree F1 at 0.7311 and the SVM F1 0.8408. The KNN suffers from the curse of dimensionality which means that for each feature there is a significant increase in resource required for computing and space.

The SVM was shown to be the most accurate out of the box with an average F1 test of 0.8408 when using a sample size of 300. While this is a good thing, the amount of time to train the model is significant. It took approx 8 times as long to train than the KNN classifier and 4x as long as the decision tree.

The Decision Tree has the lowest F1 accuracy at 0.7311 when run with 300 samples. I believe to be caused by bad out of the box parameters which could be fixed with parameter tuning. This classifier is quick to train and extremely quick to predict, being approximately 20x faster than both the KNN and SVM classifiers. This classifier also takes up very little space requirements and also has the special feature that once trained, it can identify which features are the most important.

After reviewing I have chosen the decision tree to parameter tune, as I think it cost effective, fast and should perform well once tuned.

Description: Prediction with a decision tree

The decision tree is a graph of binary questions which interrogate the input variables and leads to an outcome of either "passed" or "failed".

It is learned by splitting the training data into subsets using different algorithms which successively split the data into "passed" and "failed" groups. The better the algorithm splits the data, the more effective and important it will be in the decision tree.

The value of how good an algorithm splits the data is known as “information gain”. This is defined by how messy the data was before the algorithm, and how well sorted into “pass” and “failed” outcomes it becomes afterwards.

Final F1 Score

After parameter tuning the decision tree classifier with GridSearchCv using 3 folds, I received an F1 score of 0.8169, which is a distinct improvement over it’s untuned score of 0.7311.

I tuned 2 parameters, max_depth from 1-5 and min_samples_split from 1-15.

GridSearchCv selected the best parameters to be min_samples_split of 1 and max_depth of 1.

Supplementary

After parameter tuning the decision tree, the grid search choose the best max_depth of 1 which indicates that there is 1 feature that stands out as the major influence of a pass or fail grade. We can use the decision tree classifier as a tool to identify this feature. (Code in notebook)

Feature Importance max_depth 1
1.000 - failures

Feature Importance max_depth 2
0.717 - failures
0.142 - schoolsup
0.141 - absences

Feature Importance max_depth 3
0.497 - failures
0.196 - absences
0.138 - schoolsup
0.097 - goout
0.071 - studytime

So in summary the school should focus on the students who have failed before, as this is the key indicator whether the student will pass or not.

Feature Reduction

KNN and SVM suffer from the curse of dimensionality, and if we can reduce the number of features we should be able to get better results from these classifiers.

I chose to use the 3 most important features taken from the above tuned decision tree at max_depth of 2 to see the effects.

To do this I averaged the f1 accuracy on tuned classifiers randomizing the data each time before tuning. The code is available in “supplimentary.py”

Features chosen: ‘failures’, ‘schoolsup’, ‘absences’

Number of training samples: 300

Number of times to run: 100

	F1 Accuracy 30 features	F1 Accuracy 3 features
DecisionTreeClassifier	0.8169	0.8451

KNeighborsClassifier	0.7660	0.8742
SVC	0.8272	0.8701

The table shows the accuracy when using 3 features is distinctly higher than when using 30, especially with the KNN classifier.