

1. Classification vs Regression

This is a Classification problem since our labels have discrete values (either yes or no).

2. Exploring the Data

Facts about the dataset:

- 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

- Processed feature columns: 48
- Training set: 300 samples
- Test set: 95 samples

4. Training and Evaluating Models

Linear Support Vector Classification (SVC)

General application

Used to classify data by separating points of different classes using hyper planes. Some of the main applications of this method are

- Text and hypertext categorization
- Images classification

Strengths

- It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- It is versatile: different *Kernel functions* can be specified for the decision function.

Weaknesses

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

Reason I choose it?

I will try to apply this Algorithm because of its memory efficient. At this point, I am not sure if there exists any linear relationship between features and labels, but applying a linear classifier may help me show evidence of an existing one.

Performance table

	Training set size		
	100	200	300
Training time (secs)	0.013	0.022	0.028
Prediction time (secs)	0.000	0.000	0.001
F1 score for training set	0.826	0.831	0.838
F1 score for test set	0.672	0.791	0.778

K-nearest Neighbors

General application

This method classifies new data base on the labels of their neighbors. Some of its main applications are

- pattern recognition
- statistical classification

Strengths

- The training cost is minimal

Weaknesses

- Computationally expensive to classify any new data point because we need to compute the distance to other
- It is not easy to determine the appropriate distance to compute (euclidean, manhattan, etc...)
- Also needs to determine the correct value for K (the number of neighbors)

Reason I choose it?

Despite its bad prediction time, this algorithm might perform well since we don't have a huge training set (at most 300 training set size). K-nearest neighbors algorithm is reported to have been successful in a large number of classification and regression problems despite its simplicity.

Performance table

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001

Prediction time (secs)	0.003	0.003	0.003
F1 score for training set	0.797	0.857	0.872
F1 score for test set	0.707	0.712	0.748

Decision trees

General application

Classify the new data points by applying classification rules inferred from the training set.

Strengths

- Simple to understand and to interpret
- Able to handle both numerical and categorical data
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic

Weaknesses

- Decision-tree learners can create over-complex trees that do not generalise the data well(overfitting)
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

Reason I choose it?

It might be interesting to explain, using simple rules, why a student is predicted to fail. The explanation might help the administrators and educators in taking effective actions to avoid failure.

Performance table

	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.002
Prediction time (secs)	0.000	0.000	0.000

F1 score for training set	1.0	1.0	1.0
F1 score for test set	0.678	0.742	0.727

5. Choosing the Best Model

The linear SVC model with 200 training size has the best F1 score (0.791). But it takes much longer (0.022 secs) to train than the other models (0.001 secs).

The best model using nearest neighbors has an F1 score of 0.778 with a training size of 300. It is not efficient when it comes to predictions (0.003 secs against 0.000 for other models). Since we build the model once and use it much time for prediction, I will discard that model because of its ineffectiveness in predictions.

Decision trees on the other hand seems to have good F1 score on the on the training set (1.0), but do not perform well on the test set (0.742 for the best one). This is a symptom of overfitting.

Based on the previous analysis, I will keep the **linear SVC** model with 200 training size for its good F1 score and prediction time.

How it works?

Given a set of students (training data), linearSVC will try do draw a line to separate passing students from failing ones. That line has two main purposes:

- Maximize the margin (the distance from the line to the closest students)
- Minimize the errors (the number of failing students falling in the passing side and vice-versa)

The layout of students is based on their characteristics (sexe, age, family size, absences etc..). That is, students having similar characteristics will be placed near each other before drawing the line.

Having that line and given a new student, he will be predicted to fail is he falls in the failing side of the line. That is, if his characteristics are closer to failing students than passing ones.

Final F1 score: **0.791**