

MLND - P2 - Vishakh Rayapeta

- [1\) Classification vs Regression](#)
- [2\) Exploring the Data](#)
- [3\) Preparing the Data](#)
- [4\) Training and Evaluating Models](#)
- [5\) Choosing the Best Model](#)

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 project is a **supervised classification** problem.

Classification is the process of training a set of features to discrete output(s) or label(s)... also called classes.

Regression is the process of training a set of features to a continuous output or value.

In this project, we need to identify students that need early intervention to prevent failure. This is achieved by training a model that makes a predictive decision that is discrete with two possible output values... Will the student pass or fail the final exam.

2) Exploring the Data

Can you find out the following facts about the dataset?

- 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

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.

Steps executed in IPython Notebook.

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: Support Vector Machines

Application

SVMs are a set of supervised learning methods applicable to classification, regression & outlier detection.

Advantages of SVMs are:

- Effective in high dimensional spaces.
- Can be used when number of dimensions is greater than number of samples.
- Memory efficient: Subset of training points (support vectors) used in decision function.
- Versatile: Customizable kernel functions (linear, polynomial, rbf, sigmoid, other). This provides a mechanism for the knowledgeable user to provide domain knowledge to the model.

Disadvantages of SVMs are:

- If the number of features is much greater than the number of samples, the method is likely to give poor performances.
- Fitting time has a quadratic relationship to number of samples... this will lead to slower performance when samples > 10,000.
- Kernel hyper parameters (C, gamma) require tuning.

SVMs are a good choice for this problem because:

- Number of samples is relatively small (~300) and exceeds the number of features (~30+).
- We can attempt fitting using a choice of kernel functions to explore a larger solution space.

Performance Metrics

Support Vector Machines	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	0.904	0.890	0.876
F1 score for test set	0.768	0.797	0.784

MODEL 2: Decision Trees

Application

Decision trees are a supervised learning method that create models by learning simple decision rules inferred from the features in data samples. They are used to solve classification & regression problems.

Advantages of Decision Trees are:

- Easy to understand, interpret and visualize.

- Data prediction is fast - Logarithmic to number of data points used in the tree.
- Can produce multiple outputs.
- Do not rely on linear relationship between features.

Disadvantages of Decision Trees are:

- Prone to overfitting... Require methods like pruning, setting of minimum number of samples at leaf node or maximum depth to avoid this problem.
- Small variations can lead to major differences in tree structure... Mitigated by using trees in an ensemble.
- Practical decision tree algorithms are heuristic based and tend to optimize locally... Mitigate using ensemble learning where features and samples are randomly sampled with replacement.
- Some concepts are harder to implement: XOR, parity or multiplexer.

Decision Trees is a good choice for this problem because:

- It will allow the result to be understood and interpreted easily.
- There is no requirement of a linear relationship between features.
- It can lead to fast prediction times.

Performance Metrics

Decision Trees	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	0.779	0.742	0.672

MODEL 3: K Nearest Neighbors

Application

Nearest neighbors is an instance based learning method that is applicable to both classification and regression problems. This method does not construct an internal model but stores the instances of the training data. Classification is computed using a voting scheme of the nearest neighbors of each query point.

Advantages of K Nearest Neighbors are:

- Simple implementation with zero cost of learning.
- Highly adaptive since the algorithm uses local information with simple approximation.

Disadvantages of K Nearest Neighbors are:

- Large storage requirements.
- Computationally expensive when dataset is very large.
- Susceptible to the curse of dimensionality... Optimal sample count grows exponentially with number of features. Not suitable when the number of features is very large.
- Model cannot be interpreted... no description of learned concepts.

K Nearest Neighbors is a good choice for this problem because:

- It has a simple implementation with no cost of learning... saving compute time for learning.
- Algorithm does not make any assumptions about the features and their relationships.

Performance Metrics

K Nearest Neighbors	Training set size		
	100	200	300
Training time (secs)	0.001	0.001	0.001
Prediction time (secs)	0.001	0.002	0.002
F1 score for training set	0.880	0.879	0.881
F1 score for test set	0.772	0.779	0.780

5) Choosing the Best Model

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.

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).

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?

Choosing the Optimal Model

Of the three models selected, Decision Trees exhibit the problem of overfitting. The Decision Trees model achieves a perfect F1 score (1.0) for all three training set sizes, however the test set F1 scores (0.779, 0.742, 0.672) are considerably lower. It is possible to use ensemble methods such as Random Forests, AdaBoost to address overfitting. However, this will involve computational overhead.

Support Vector Machines and K Nearest Neighbors both exhibit similar F1 scores of ~0.780 and prediction times of 0.002 seconds. However, the K Nearest Neighbors has one advantageous attribute: Simple implementation with zero learning computation overhead.

This makes **K Nearest Neighbors** the best choice amongst the three models.

Describing the Model in Layman's Terms

The 'K Nearest Neighbor' method uses the existing student records database as a lookup table resource. When a predictive decision for a new student needs to be made, the model performs a lookup on the records database. The model will search for and locate the records of other students that share the closest attributes to the student being queried. The student records that share the closest attributes to the new student are called 'neighbors'. The outcome for the new student is then determined using a voting scheme of the nearest neighbors.

This model does not require fitting or training of a complex model. This saves computation overhead time typically required for 'training' other types of machine learning algorithms.

Model Tuning

Using GridSearchCV, the following three parameters were tuned

- n_neighbors : Number of neighbors... provide a set of values to search in.
The default '5' is low for the number of features in this problem.
- weights : Uniform weights for all neighbors or weigh them inversely by distance.
- p : Choice of manhattan, euclidean or minkowski distance.

Tuned F1 Score

F1 score for training set: 0.818

F1 score for test set: 0.784