**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 supervised classification prediction problem. Classification deals with a discrete set of output labels, such as ‘pass’ or ‘fail’. The objective of a supervised classification problem is to learn about the relationship between inputs and discrete outputs (e.g. pass or fail) and then predict these discrete outputs on new data. Regression is not useful for this exercise as it is more suited for problems with continuous values, such as 5.9 or 9.4.

**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: 30

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

**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.*

1. Decision Tree Classifier

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

The decision tree classifier, as the name implies, is a classification algorithm. This classifier is relatively straightforward and easy to understand (e.g. it creates a downward tree structure). One of it’s most well known strengths is it can be visually graphed.

One of the decision tree’s greatest weaknesses is that it is prone to overfitting out-of-the-box. Tweaking is required to stop the growth of the tree at the appropriate time. I’ve also learned that decision trees create biased trees if some classes dominate.

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

I chose this model as it’s a simple model to understand and it handles both numerical and categorical data (although the categories for this data set have been transformed to 1’s and 0’s).

Looking at the data, I would think that the ‘gini’ criterion will identify the best feature to split the data early-on to create the shortest tree possible. I believe features, such as previous course failures (failures), will be split earlier within the decision tree process.

* *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](https://docs.google.com/document/d/1Goxw-6M0umOCokCFqTr-g7SU_5f6MIm_wqXmh9Cnjhw/pub" \t "_blank) showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.*

|  |  |  |  |
| --- | --- | --- | --- |
| DecisionTreeClassifier | Training set size | | |
| 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.002 | 0.005 |
| 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 | 0.765625 | 0.6446280991 | 0.7230769230 |

1. K Neighbors Classifier

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

K Neighbors Classifier makes its prediction based on other points of data that are closest to the target. The “K” is the number of neighbouring labels to include when predicting the target label. The “K” is setup by the user. This algorithm is known as a lazy learner. In other words, it does all of it’s calculation not in training, but in querying/ predicting the target label. This algorithm simply stores the data during the training phase.

The greatest strength of this algorithm is in its simplicity where a prediction is computed from a simple majority vote of the nearest neighbors of each point. In addition, new training examples can be easily added during training as it is a lazy learner; a lazy learning algorithm is one that delays generalization until a query is executed. The training phase in this algorithm is less resource-intensive than eager learning algorithms.

Some of the challenges in using a nearest-neighbor algorithm is that it is known to be expensive and slow when querying the data for a prediction. This is because the prediction is calculated by churning through the data as opposed to coming up with an algorithm/ function created by historical data. This is not a problem with smaller datasets but is noticeably more resource-intensive with larger sets of data.

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

I chose to test out this model as I feel that K-NN will do a good job in correctly labeling target students based on previously scored students with “neighbouring” data. My main concern is the processing power (memory) required during query time.

* *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](https://docs.google.com/document/d/1Goxw-6M0umOCokCFqTr-g7SU_5f6MIm_wqXmh9Cnjhw/pub" \t "_blank) showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.*

|  |  |  |  |
| --- | --- | --- | --- |
| KNeighborsClassifier | Training set size | | |
| 100 | 200 | 300 |
| Training time (secs) | 0.000 | 0.001 | 0.001 |
| Prediction time (secs) | 0.002 | 0.005 | 0.009 |
| F1 score for training set | 0.8321167883 | 0.8501742160 | 0.8452655889 |
| F1 score for test set | 0.791176470 | 0.7714285714 | 0.8085106382 |

1. Support Vector Classifier

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

A Support Vector Classifier belongs to the Support Vector Machine family of algorithms. These sets of algorithms can be used for linear and non-linear classification problems. SVMs work best when there is a clear margin of separation between classes. A kernel function can be used to solve non-linear problems by transforming them into linear problems. This is said to be using higher-order parameters.

Support vector algorithms have been successful in many real-world problems such as text (and hypertext) categorization, image classification, bioinformatics (Protein classification, Cancer classification), and hand-written character recognition.

A support vector algorithms are known to work well when there is a clear margin of separation between classes. Another known strength is that the algorithm still works well when there is a high number of dimensions and the sample size is small. It also pays less attention to outliers as it only uses a subset of the training data.

Support vector algorithm may struggle when very large datasets are used or when there is a lot of pollution in the data (noise in the data). Support vector algorithms are also known to be prone to overfitting out-of-the-box. Tweaking parameters are required to circumvent this challenge.

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

The SVC algorithm looks to be a good fit for the data as it’s data is rather small in size. If any outliers exist within the data, SVC is designed to pay less attention to those outliers.

* *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](https://docs.google.com/document/d/1Goxw-6M0umOCokCFqTr-g7SU_5f6MIm_wqXmh9Cnjhw/pub" \t "_blank) showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.*

|  |  |  |  |
| --- | --- | --- | --- |
| SVC | Training set size | | |
| 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.004 | 0.012 |
| Prediction time (secs) | 0.001 | 0.003 | 0.010 |
| F1 score for training set | 0.8951048951 | 0.878688524 | 0.8596112311 |
| F1 score for test set | 0.8129032258 | 0.8205128205 | 0.8235294117 |

1. GaussianNB

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

GaussianNB is part of the Naïve Bayes family of algorithms. Naïve Bayes algorithms have been used to efficiently identify spam emails and fraudulent activity on Visa cards. It is known to be very effective in text classification problems.. This algorithm is know to work well in large datasets with a lot of noise. It is fast to train and classify; this is because the algorithm assumes that features are independent of each other. It is not sensitise to irrelevant data.

Naïve Bayes algorithms are known to be weaker with smaller datasets. The algorithm is also known to assume independence of features; this is an asset when dealing with large and noisy data but it becomes a double-edged sword when dealing with a smaller and more predictable datasets.

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

Honestly, I chose a Naïve Bayes algorithm because I keep hearing it’s the gold standard in machine learning. I feel like this algorithm had to be included in the top four of my choices for this exercise (although this assignment only asked for three algorithms).

But if I can try to answer this question, I would say that given the data, a Naïve Bayes algorithm would not be impacted by irrelevant features (if irrelevant features exist in the data). I also chose this algorithm in the case where the dataset would contain a lot of noise in the data; a Naïve Bayes algorithm still works well with noisy data.

* *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](https://docs.google.com/document/d/1Goxw-6M0umOCokCFqTr-g7SU_5f6MIm_wqXmh9Cnjhw/pub" \t "_blank) showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.*

|  |  |  |  |
| --- | --- | --- | --- |
| GaussianNB | Training set size | | |
| 100 | 200 | 300 |
| Training time (secs) | 0.001 | 0.001 | 0.001 |
| Prediction time (secs) | 0.000 | 0.000 | 0.000 |
| F1 score for training set | 0.4337349397 | 0.8243727598 | 0.8243727598 |
| F1 score for test set | 0.4222222222 | 0.7883211678 | 0.7883211678 |

**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 my analysis, I applied four algorithms to the dataset; Decision Tree Classifier, K Neighbors Classifier, Support Vector Classifier, and Gaussian Naïve Bayes. If we based the decision strictly on the algorithm that produced the highest predictive success, the winner would be the Support Vector Classifier with a predictive success rate of 82% (f-1 score in technical terms). However, the Support Vector Classifier was also the most resource-intensive algorithm out of the group of four. Based on the criteria of a relatively small dataset, the wish to keep resources and cost to a minimum without sacrificing performance, the K neighbors classifier algorithm, in my opinion, is the most appropriate model. Compared to the other three algorithms, the K Neighbors Classifier produced a predictive success of 80%, while keeping system resource requirements to a minimum. I recommend this algorithm with caution as another algorithm will be better-suited if the criteria changes to accommodate a larger dataset.

In conclusion, the K neighbors classifier algorithm is the most appropriate model based on the available data, limited resources, cost and performance. The K neighbour classifier performs well with smaller data sizes (395 records in this case), without asking too much on memory and other resources, resulting in an effective and low cost solution.

*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).*

The most appropriate model for identifying students that may need early intervention is the K Neighbors Classifier. This solution, based on the criteria of a) available data, b) limited resources, c) cost and d) performance, performed better than the other 3 algorithms chosen for this analysis: Decision Tree Classifier, Support Vector Classifier, and Gaussian Naïve Bayes.

So how does the K Neighbors Classifier algorithm work you may ask? Similar to selling a house in a neighbourhood, with all else equal, the house will have the same value as the other houses nearest to it. Tying this back to this assignment, the K Neighbors Classifier algorithm assigns a target label of “pass” or “fail” to every queried data point. The pass or fail label for each of these data points is done by majority vote. In other words, a pass label is given if the majority of the historical data points surrounding the queried data point is linked back to the student passing their final exam.

The “K” in the equation represents the amount of data points to consider for the “voting” process. In my analysis, the best predictive score included the nearest 7 data points for each of the data points. In other words, the 8th nearest data point was not considered in the “pass vs. fail” voting process.

*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.*

parameters = {'leaf\_size':(5,30),'n\_neighbors':(5,6,7),'weights':('uniform','distance')}

*What is the model’s final F1 score?*

|  |  |
| --- | --- |
| KNeighborsClassifier | Training set size: 300 |
| Testing set size: 95 |
| Training time (secs) | 0.238 |
| Prediction time (secs) | 0.004 |
| F1 score for training set | 0.8259860788 |
| F1 score for test set | 0.8251748251 |