Student Intervention Project Report

1. Classification vs. Regression

This is a classification problem because it is a yes or no question. In regression, we map our input to a real number. For example, if we wanted to find how likely the person is going to graduate in percentage, and want the prediction to be a continuous value, then we would consider this as a regression problem. However in classification, our goal is to label the input to different categories which is usually a small discrete set. In this case, we are trying to figure out if a student needs an early intervention or not (2 classes).

2. Exploring the Data

a. Total number of students

i. 395 students

b. Number of students who passed

i. 265 students

c. Number of students who failed

i. 130 students

d. Graduation rate of the class (%)

i. 67.09%

e. Number of features

i. 30 features

3. Preparing the Data

shown in code

4. Training and Evaluating Models

Choose 3 supervised learning models that are available in scikit-learn, and appropriate for this problem.

a. Model One: Non linear kernel Support Vector Machine

i. What is the theoretical O(n) time & space complexity in terms of input size?

The running time can be more than $O(n^2)$. Training can become extremely slow with 10,000 samples. Space complexity is $O(n^2)$.

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

Support Vector Machine can be used for both classification and regression problems. For classification, general applications can be text categorization or image classification.

Strengths

- Works well in high dimensional spaces even if there are more dimensions than samples.
- Memory efficient for using support vectors.
- Using a nonlinear kernel trick, you can capture a more complex relationship between the datapoints. You will not have to do the difficult transformations yourself.

Weaknesses

- For a large dataset with a lot of noise, SVMs will be slow and possibly produce overfitting

- If there are too many features than samples, SVM will perform poorly

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

I believe a non linear SVM will work better than a linear SVM because there are 30 features and based on that the data will not be linearly separable. Also Non Linear SVM works well for a high dimensional data like the student data.

iv. Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

	Training set size		
	100	**200**	**300**
Training time (secs)	**0.001**	**0.004**	**0.010**
Prediction time (secs)	**0.001**	**0.003**	**0.006**
F1 score for training set	**0.852**	**0.8607**	**0.842**
F1 score for test set	**0.789**	**0.8026**	**0.797**

b. Model Two: Random Forest

i. What is the theoretical O(n) time & space complexity in terms of input size?

O(K(mnlogn)) where K is the number of trees, m the number of features, and n the input size.The space complexity is $O(K(2^x - 1))$ where K is the number of trees and $2^x - 1$ nodes for each tree where x is the max depth.

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

Random Forest is an ensemble of decision trees. Some general applications could be predicting weather or deciding to go to a specific restaurant or not. Random forest is mostly applied to answer classification problems. Random Forest is usually not great for regression because it will not be able to predict answers outside of the training data range.

Strengths

- while only one decision tree is likely to suffer from high variance or high bias, random forest uses the technique to average its predictions to find the right balance.
- Random Forest corrects the decision tree's possible overfit to the training set.
- accurate and runs efficiently on large datasets

Weaknesses

- Slow predictions
- For better accuracy, need more trees. This can slow down the training performance.
- Will not work as well with a small dataset

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

Although the dataset is small, I decided to try the Random forest model. I used SVM first assuming that maybe the data is outlier free and that SVM is great for small data. However, there is a possibility that the data could contain some outliers and this can affect the accuracy. Random forest is robust to outliers because it uses a subset of training sets with bagging and a subset of features to reduce the outliers effect on the result.

iv. Using 15 trees

	Training set size		
	100	**200**	**300**
Training time (secs)	**0.012**	**0.018**	**0.024**
Prediction time (secs)	**0.003**	**0.003**	**0.007**
F1 score for training set	**0.993**	**0.996**	**0.995**
F1 score for test set	**0.78**	**0.791**	**0.76**

c. Model Three: Bagging with K Nearest Neighbors

i. What is the theoretical O(n) time & space complexity in terms of input size?

Time Complexity: O(K(mnlogn)) where K is the number of the base classifiers and O(mnlogn) to find the m nearest neighbors for every data point.

Space Complexity: O(K(n))

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

A bagging classifier is an ensemble of base classifiers that fit each classifier to a random subsets of the original data and averages the prediction of the individual classifiers to create the final output. Bagging classifier is used to increase accuracy by combining the weak learners (e.g. decision trees, knn ,etc) to provide a strong learner. K nearest neighbors is used as a base classifier in this example. General application for KNN can be computer vision or recommending systems.

Strengths

- reduces variance and avoids overfitting

- (KNN) successful in classification problems where the decision boundary is irregular

Weaknesses

- (KNN) stores instances of the training data and computes nearest neighbors of each point so running time is very slow compared to other models when predicting
- a linear combination of KNN classification can be harder to interpret than one KNN classification
- (KNN) KNN is a lazy learner. It will be slow during prediction when too many data points
- KNN will not know which attributes are more important than others because all attributes have same weight

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

Based on the features, I predicted that the decision boundary would be not linear at all and I remember that KNN is non parametric thus the decision boundary is flexible. There are not many data points to begin with so I thought the performance will not be that bad. I chose a bagging method to improve the results even further.

iv. Produce a table showing training time, prediction time, F1 score on training set and F1 score on test set, for each training set size.

	Training set size		
	100	**200**	**300**
Training time (secs)	**0.009**	**0.013**	**0.014**
Prediction time (secs)	**0.005**	**0.013**	**0.024**
F1 score for training set	**0.842**	**0.847**	**0.845**
F1 score for test set	**0.805**	**0.779**	**0.771**

5. Choosing the best model

1. 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?

Of the models I tested, I believe the first model with the Support Vector Machine Algorithm is the best model. Support Vector Machine is known to do well in high dimensional space, producing non linear decision boundaries, and due to its implementation it is memory efficient. On the contrary, it can perform badly when there are more features than samples, and if there are too many data points and noise, it will produce overfitting.

In our situation, SVM is the most appropriate. We have enough available data, not too little and not too much. The F1 Score is the highest for training size 200 compared to training size 100 and 300. Thus we can get a good performance with just 200 samples for training. We have more samples than features, thus we do not have to worry about a bad performance. The F1 score is relatively higher for training size 200 compared to the random forest and bagging model. As for cost, SVM is memory efficient and faster than the K nearest neighbor. Although the random forest model is faster in training and prediction than SVM, its performance is not as great as SVM. You can see that the random forest model does very well (almost 1 as its F1 score) during training but the score drops by more than 0.2 for prediction. This is a clear symptom of overfitting. This can be fixed by growing a larger forest but prediction will be slower.

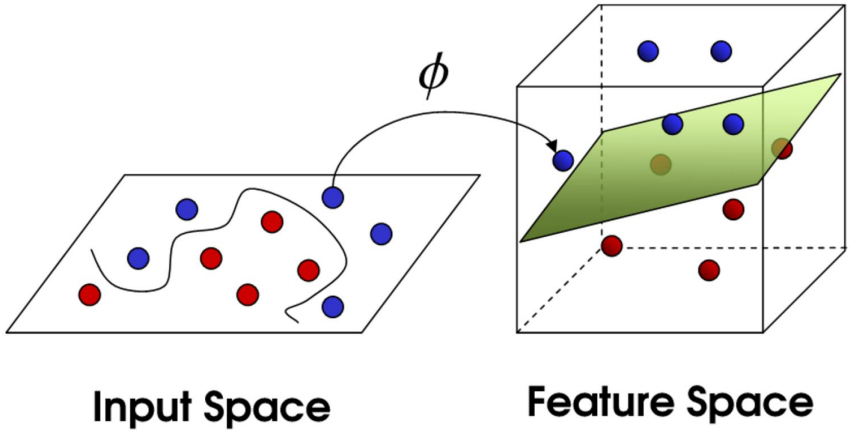
2. In 1-2 paragraphs explain to the board of supervisors in layman's terms how the final model chosen is supposed to work (for example if you chose a Decision Tree or Support Vector Machine, how does it make a prediction).

The Support Vector Machine can classify whether a student needs early intervention or not by creating a line between the students based on their background information.

Let's say students who need early intervention and who don't are all standing around on a field. They are positioned on the field based on their extracurricular activities and GPA. Our goal is to separate those students and create some kind of boundary. We can draw a straight line to accurately separate those students. However if we add new students to the field, students who need early intervention may be on the wrong side of the line vice versa. What Support Vector Machine does in this situation is to draw a better line between the two groups that has a maximum gap on either side of the line.

However if we observe other factors that could affect the student's performance in school such as their parent's occupation, the students position will be more complicated on the field. It becomes harder to separate the students with just a line. Here, SVM is able to separate them with a line but in a higher dimension.

Here in this video (https://youtu.be/3liCbRZPrZA), using a kernel trick, we can draw a line to separate the data in 3D space. SVM maps the data points in a higher dimensional. For example in the image below, to see the balls in 3D space, you can throw the balls in air. The balls are positioned in a way where if you look straight from above, the balls are in the same position as we looked at it in the 2D space. Then it slips a piece of paper (hyperplane) that can accurately separate the data. Taking all the information you obtained from the piece of paper, you convert it back to 2D space. In a 2D view, the datapoints will be divided by some squiggly line.



Source:

What does support vector machine (SVM) mean in layman's terms? (n.d.). Retrieved December 31, 2015, from https://www.quora.com/What-does-support-vector-machine-SVM-mean-in-laymans-terms

3. 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.

   Best parameters for the final tuned SVM model is {'kernel': 'rbf', 'C': 1, 'degree': 3}

4. What is the model's final F1 score?

   F1 score for test set: 0.797385620915