Project 2 - Building a Student Intervention System

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 obviously a Classification Machine Learning Problem, as we are predicting a 1 or 0 (yes or no) to whether or not a student will pass their high school final exam. This is a discrete case of only predicting a binary result, not on a continuous scale like regression.

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 (%age) 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: 0.67%

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.

- I used sklearn train_test_split to random assign values for the training and testing sets. I use this because I can set a random seed on the on train_test_split and I can easily reproduce my results. If just use numpy's shuffle or permutation methods, I will receive a different subset of data each time and this way my results are repeatable.

4. Training and Evaluating Models

SVM

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

- The standard SVM training has O(n^3) time and O(n^2) space complexities, where n is the training set size. *Core Vector Machines: Fast SVM Training on Very Large Data Sets* by Ivor W. Tsang, James T. Kwok, Pak-Ming Cheung

What are the general applications of this model?

- SVM Often used in text processing, detection of human faces in gray-level images and bioinformatics tasks including analyzing biological samples, tissue classification, gene function prediction, and protein structures. *Support Vector Machines with Applications* by Javier M. Moguerza and Alberto Munoz

What are its strengths and weaknesses?

- Strengths -

    - Can finding a hyperplane (maximum margin) that separates the data as well as possible unlike many other classification techniques
    - Most importantly can learn non-linearly decision boundaries with the use of kernels by enlarging the feature space
    - Doesn't suffer from multicollinearity
    - Works on basis of a hinge loss
- Weaknesses -
    - The major downside of SVMs is that it can be inefficient to train, memory intensive.
    - Only really directly applicable to two class tasks, unless you use one-versus-one or one-versus-all methods
    - Can also lead to overfitting when features are large
    - Parameters are not interpretable

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

- I chose a SVM to represent this data because I like like the notion of a maximum margin classifier and with a feature vector with a length of 30, it is pretty likely that the decision boundary would be nonlinear and with the default kernel parameter of 'rbf' it would seem to would work well for this data.

Predict labels (for both training and test sets), and measure the F1 score

- Understandably as the training size increases the test set f1 score increases, however not the same for the f1 score for the training set as initially it had higher variance and might have been overfitting, but we are more concerned about unseen data regardless like the test set.

SVC	Training set size		
	100	200	300
Training time (secs)	0.002	0.003	0.006
Prediction time (secs)	0.001	0.002	0.004
F1 score for training set	0.882353	0.881356	0.845987
F1 score for test set	0.774648	0.783784	0.833333

Note: Training / Prediction times might be a little different from notebook as they vary for per run

KNN

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

- From the lecture, given sorted data points(which can be done by something like merge sort or quicksort (n log n)), the running time for the learning rate of KNN is O(1) and the

query time is O(log n + k) or simply O(log n). The space of the learning rate of KNN is O(n) and the query rate is O(1). Since KNN is a lazy learning method classification time is nonlinear, training is fast however classification is slow. *The Analysis and Optimization of KNN Algorithm Space-Time Efficiency for Chinese Text Categorization* by Ying Cai Xiaofei Wang

What are the general applications of this model?

- Text categorization - instances are natural language documents, can be represented with a set of word and word weight pairs. Agriculture - simulating daily precipitations, weather variables, forest inventories, and estimating forest variables all done with satellite imagery. Finance - Stock Market forecasting, predicting the price of a stock, credit rating, financial risks. Medicine - predict whether a patient with previous heart attack, will have a second heart attack, amount of glucose in the blood, risk factors for prostate cancer. *Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background* by Sadegh Bafandeh Imandoust And Mohammad Bolandraftar

What are its strengths and weaknesses?

- Strengths -
    - The biggest strength is how simple and effective it is
    - Robust to noisy data
    - Effective with large amounts of data
- Weaknesses -
    - Poor running time when training data is huge (curse of dimensionality), high computation cost
    - Sensitive to irrelevant or redundant features as all features contribute to the similarity
    - Distance based learning is hard to determine which distance metric to use

Application of K-Nearest Neighbor (KNN) Approach for Predicting Economic Events: Theoretical Background by Sadegh Bafandeh Imandoust And Mohammad Bolandraftar

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

- It seems appropriate that a distance algorithm would work well with student data, as you would think that at risk students and intelligent students would be more likely to be in separate clusters and KNN would distinguish well between them. Also with using an F1 score as a performance metric, KNN is suppose to have good precision and recall rates.

Predict labels (for both training and test sets), and measure the F1 score

- Generally as the more data that the classifier receives to train on the better the model can be fit, therefore the higher f1 score for the test set.

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

Prediction time (secs)	0.002	0.003	0.005
F1 score for training set	0.824427	0.809859	0.849057
F1 score for test set	0.751880	0.753623	0.794521

Note: Training / Prediction times might be a little different from notebook as they vary for per run

Tree Ensemble - Random Forest

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

- For an unpruned *decision tree* is O(mtry * n log(n)), where n is the number of objects and mtry is the number of attributes, therefore building a random forest with ntree, the runtime of random forests is O(ntree * mtry * n log(n)) where ntree is the number of trees, mtry is the number of variables randomly sampled as candidates at each split and n is number of objects. The instance space is R^ntree for the number of trees.

Random Forest Based Feature Induction by Celine Vens and Fabrizio Costa

What are the general applications of this model?

- Random Forests can be applied in a broad spectrum of biological tasks, often used medical fields, such as diagnosis diseases or cancer or used for tissue classification *Random Forest Based Feature Induction* by Celine Vens and Fabrizio Costa

What are its strengths and weaknesses?

- Strengths -
    - Can use non linear features or even features that interact linearly
    - Possibility to evaluate features importance, give you a really good idea of which features in your data set are the most important
    - Handles high dimensional spaces and a large number of training examples
    - Scalable, don't have to tuning a bunch of parameters like you do with SVM
- Weaknesses -
    - Can over-fit data sets that are particularly noisy
    - Slow to create predictions once the model is trained since more accurate ensembles require more trees, therefore model then becomes slower.
    - Compared to simple decision tree, RF are far less comprehensible

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

- I chose to use a random forest model because originally I wanted to use a decision tree; however, simple decision trees don't have as much predictive power and a random forest is just an ensemble of random decision tree classifiers by making predictions from combining predictions of the individual trees. And although decision trees are more interpretable, random forest have feature importances where the school if desired can determine the important factors that contribute to success or failure of a student.

Predict labels (for both training and test sets), and measure the F1 score

- I just threw a couple parameters into the model of (n_estimators=200, max_depth=10) as they typically the more trees the better the performance, however didn't use gridsearch. Here it is obvious that the model is memorizing the training data and overfitting. But with the higher amounts of training size the testing f1 score increases.

Random Forest	Training set size		
	100	200	300
Training time (secs)	0.137	0.115	0.118
Prediction time (secs)	0.009	0.011	0.013
F1 score for training set	1.000000	1.000000	1.000000
F1 score for test set	0.786207	0.786207	0.863014

Note: Training / Prediction times might be a little different from notebook as they vary for per run

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?

Based on the available data, limited resources, budgets, and performance of each algorithm, if the school district has a goal to reach a 95% graduation rate, the best model in my opinion is the Support Vector Machine. Although the typical SVM has a training time of O(n^3) and by looking at the training and prediction times, it looks as if the model training times are doubling as the training size increases by 100. However, in contrast to the other models, I believe the Random Forest training and prediction time is way to high for an increased amount of data and the KNN model just doesn't predict as well as desired. In this situation it is much more important to have higher predictive power in something like a school and identifying students who need intervention.

The chosen model, a Support Vector Machine is very unique. I will not go into the mathematical details of the algorithm but a support vector machine is a generalization of a simple and elegant classifier called the maximal margin classifier. SVMs use an optimal separating hyperplane to distinguish between the classes. For example in three dimensions, a hyperplane is a flat two-dimensional subspace separating the classes. The intuition behind a

SVM is that there often is multiple hyperplanes that separate the data, but with an SVM, this unique model chooses the ‘best’ hyperplane where the margin is largest, ie. the farthest minimum distance from the training observations. Then the prediction is simple, we then can classify a new observations (new students) based on which side of the maximal margin hyperplane that student lies. The cool and highly powerful thing about SVMs is that the classes don’t have to be separable by a single linear line or hyperplane. But, they can incorporate in something called the kernel trick, where we can convert a linear classifier into one that produces non-linear decision boundaries by altering parameters and essentially fitting in a higher-dimensional space. This kernel allows the model to gain flexibility to and greatly increase accuracy and predictive power. The final F1 Score using StratifiedShuffleSplit for the CV and GridSearchCV for tuning with all of the data is 0.977860.

An Introduction to Statistical Learning by Trevor Hastie Robert Tibshirani