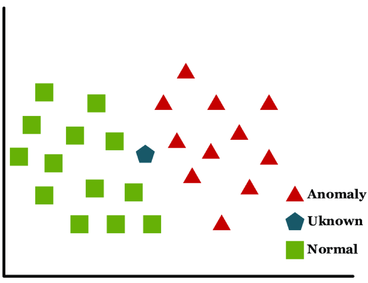
Model Comparisons and Analysis

by Alejandro Capecchi and Joon Park

Index:

K Nearest Neighbors

Abbreviated as KNN, the K Nearest Neighbors model is a highly flexible classification model. It performs classification by computing the distances between a **query** point and its K nearest neighbors in P dimensional space, P being the number of features. In simpler terms, it finds the nearest K points and takes a tally of their classes, then predicting this new query point to be of the majority class. It is also a lazy learner meaning that it does not fit any training data during the training stage. It merely stores it for when predictions are needed. Because of this, it is an extremely fast algorithm for prediction.



We say that KNN is flexible because it's decision boundary is solely determined by the training data. This flexibility is determined by the K parameter, the number of neighbors. The smaller K is, the more wiggly the boundary is and the more it tends to overfit since it only finds 1 neighbor. As K grows, this boundary becomes more rigid and robust to random noise. Thus, to find the best KNN model we iterate through a range of K's and determine which had the most accuracy.

Additionally, points can be weighted by distance for different results. The ones involved were uniform and inverse distance, the former treating all distances as equal (hence uniform) while the latter penalizes points that are very far away from the query point.

Using all 10 features, it was found that on average:

1. 86.6% Accuracy, Inverse Distance Weights, K=5
2. 85.9% Accuracy, Uniform Weights, K=5

Using only the 4 numerical features, it was found that on average:

1. 71.4% Accuracy, Uniform Weights, K=13)
2. 67.7% Accuracy, Inverse Distance Weights, K=2

All models were validated with LOOCV.

Aggregating all 10 variables had the best results on predicting whether a student should or should not go to college, with inverse distance slightly boosting predictive accuracy. Limiting the feature size to 4 severely hampered the accuracy of the models. So if using KNN, it is best to stick to all 10.

Linear and Quadratic Discriminant Analysis

- ### Linear Discriminant Analysis:

The first of what we call generative models, LDA is an older method of classification. First we assume that the P predictors form a vector X which comes from a multivariate Gaussian distribution. Each individual predictor is assumed to be approximately normal and to have some correlation between one another. These correlations are stored in Σ, the correlation matrix. Classification is done by inputting the estimated values into the function for each of the K classes. Whichever class maximizes this value is the most likely one according to LDA.

$$\delta_k(x) = x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma^{-1}\mu_k + log(\pi_k)$$

Σ = correlation matrix

x = vector of predictors (size p)

μ_k = mean of class k

It is important to note that this is a linear function and hence has a linear decision boundary, making it a poor fit for more rough data.

No required parameter tuning makes this easy to optimize, with Scikit providing three solving methods of which only two were used: Singular Value Decomposition and Eigenvalues. Leidot-Wolf shrinkage, a dimensionality reduction algorithm, was also available when using the eigen solvers. Here are our reported findings.

- ### Quadratic Discriminant Analysis

Sharing many similarities with LDA, QDA differs in that it assumes each class K has its own Σ_k covariance matrix rather than a single one. This is useful when the LDA assumptions fail, but is generally not as good. However, it requires many more parameters to compute and can thus be more expensive. Classification is done the same way, whichever class maximizes the function:

$$\delta_k(x) = -\frac{1}{2}x^T\Sigma_k^{-1}x + x^T\Sigma^{-1}\mu_k - \frac{1}{2}\mu_k^T\Sigma_k^{-1}\mu_k - \frac{1}{2}log|\Sigma_k| + log(\pi_k)$$

LDA MODELS:

Using 10 features:

1. 86.2% Accuracy with svd

Using 4 features:

1. 81.3% Accuracy with eigen and Leidot-wolf shrinkage
2. 81.1% Accuracy with svd

Using 3 features (excluding parent_age):

1. 80.8% Accuracy with eigen and Leidot-wolf shrinkage
2. 80.6% Accuracy with svd

Interestingly, the simple 3 feature model (parent_salary, house_area, and average_grades) has very good accuracy and is very simple. Leidot wolf shrinkage also helped during testing, as eigen solvers became better with only numeric variables in play.

QDA MODELS:

Using 10 features:

1. 59.6% Accuracy.

Using only the 4 numerical features:

1. 82.6% Accuracy.

Using only the 3 numerical (excluding parent_age) features:

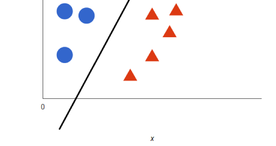
1. 81.9% Accuracy.

Notably outperformed LDA in all manners but the full 10 feature fit of the model. This is because of a dire violation of its assumptions of covariance matrices by the categorical variables.

Support Vector Machines

Abbreviated as SVMs, this method seeks to split the data with the most separation possible via a hyperplane. In 2D space, this is simply a line, in 3d this is a plane, in 4d it would be a 4d hyperplane, and so on. How this is achieved is by finding some number of points from each class that are the nearest to each other and a hyperplane is drawn through their midpoint. This becomes unachievable when some classes are mixed in with others, and thus a parameter that allows some error is allowed. It widens the margins and allows a fit.

They are expensive to recompute and were thus not a top choice, however they did have good metrics on this dataset and tend perform well when there are many more predictors than observations. (P>>N)



SVM MODELS

Using all 10 variables the SVM (LinearSVC) had an accuracy of 85.9%.

Using all 4 numeric variables the SVM (LinearSVC) had an accuracy of 49.5%

Random Forest

The random forest model is a model made up of multiple different decision trees. Based on the prediction made by the number of trees, the models take a voting system and makes one single prediction. This machine learning model generates a tree like structure based on the the gini index (measure of variance). The higher the feature is on the tree means that there is lower value of gini index. This means that there is less variance and therefore the model is considered more important.

The random forest model contains many different parameters which affect the tree model. One of the important features is 'max_features' which determines how many features the model will consider. One could take all features, take 20% of the features or take the square root of the number of features. Depending on the number of features, there needs to be a balance between computation speed and the diversity within each of the trees. Too little balance means trees look similar in structure and vice a versa. There are other parameters such as the max_depth (which controls the depth of the tree) and min_samples leaf which control the size of the trees and help with overfitting.

For our college dataset, these were the list of features which produced the accuracy of 86.5% with cv equal to 5 where each parameter determines the shape of the overall trees:

- 'fselection__k': 8,
- 'poly_features__degree': 1,
- 'random_forest__max_depth': 7,
- 'random_forest__min_samples_leaf': 2,
- 'random_forest__min_samples_split': 5

Through this model, we were also able to determine which features the model thought were the most important. The most important feature is the student's average grades in determining whether or not the they will go to college.

Logistic Regression

The logistic regression model is classification method used to calculate the probability of an event (in this case, binary event) of whether that event will happen. This is because the value of the predictions are between 0 and 1. The logistic regression model estimates each feature's beta coefficients and uses maximum likelihood estimation to reduce errors and optimize the best values from the training data. This, in turn, generates the probabilities and allows the model to generate predictions.

After using GridsearchCV with the CV of 5, we were able to find the best parameters for Logistic Regression with the values of:

- 'fselection__k': 7
- 'logistic_regression__C': 100
- 'poly_features__degree': 1

As shown, it chose 7 features and resulted in a test accuracy of 84%. It is interesting to note the large C parameter value of 100 as it means that there isn't much penalty applied to the model in training and thus will not regularize the beta parameters discussed earlier. There are other important parameters to tune such as penalty and solver. However, for the purpose of this dataset, there weren't any differences in terms of speed or accuracy.

Conclusions

While all models had pretty good performances, there are many winners in categories and some things of note.

- Average grade was the single most important variable for predicting whether a student will go to college or not.
- With a fully fitted 10 feature model the top three models are:
  1. KNN, 86.6%
  2. RF, 86.5%
  3. LDA, 86.2%
- Logistic regression had a competitive 84% accuracy with only 7 predictors
- The most robust models with only 4 predictors:
  1. QDA, 82.6%
  2. LDA, 81.3%
  3. KNN, 71.4%

In terms of cost-efficiency, KNN is not only the best simplest but the best predictive model if considering all variables. If considering the most parsimonious model, then QDA triumphs over all with a sutnning 82.6% and only 4 predictors.