CS 324 Data Mining **Homework #3**
Il Shan Ng **April 17, 2017**

1 Introduction

In this report, I describe how I implement a Naive Bayes classifier to predict whether a writing portfolio will be given a "needs work" based on a set of features. I begin with exploratory analysis in §2 to understand the distribution within each feature, and to decide whether to model each feature as taking on discrete or continuous values. In §3, I discuss whether some features may be omitted based on whether they contribute any information to the model. In §4, I present my classifier's accuracy and show how it changes with different modeling decisions. Finally in §5, I compare the Naive Bayes classifier to the *k*-nearest neighbors classifier, and discuss the pros and cons of both algorithms.

2 Exploratory Analysis

The data set contains 141 training instances, each of which has 13 features that may be classified as discrete or continuous as shown in Table 1. Abbreviations for the features' names are given in parentheses. Each training instance also has a ground truth label with the classes "needs work" (`+1`) and otherwise (`-1`).

Table 1: Discrete and Continuous Features in the Data Set

Type	Features
Discrete	whether the student is from Minnesota (`mn`)
	whether the student is international (`int`)
	student's birth year (`birthyear`)
	number of essays submitted (`essays`)
Continuous	student's verbal SAT score (`verbal`)
	student's math SAT score (`math`)
	student's cumulative GPA (`gpa`)
	number of abroad credits (`abroad`)
	number of AP credits (`AP`)
	number of CS credits (`CS`)
	number of English credits (`english`)
	number of science credits (`science`)
	number of writing credits (`writing`)

Each of the discrete features takes on only a few possible values, such that the counts for each value are sufficiently large for proportions to be computed accurately. The continuous features, on other hand, have to be pre-processed in one of two ways: abstracting the feature values to a known probability distribution, or discretizing them by binning them into factor levels. I now look at summary statistics to justify treating the first four features as discrete, and produce exploratory plots to determine the correct way in which to pre-process each continuous feature.

2.1 Discrete features

Although `mn` and `int` were coded in the data set as binary 0/1 integers, it is obvious that these two features are discrete because intermediate values between 0 and 1 would be make sense. By similar reasoning, `birthyear` should be treated as a discrete feature since the years, although numerical, are really factor levels whose interpolations have no interpretation. Table 2, which displays the counts for the different years in `birthyear`, shows that there are generally sufficient counts for each year so that no grouping of levels is necessary. Although there is only one student born in 1980, the use of Laplace smoothing in the algorithm ensures that the estimated conditional probability for this student will not be zero.

Table 2: Counts for Different Years in `birthyear`

Birth year	1980	1982	1983	1984	1985
Count	1	15	52	45	27

Table 3 shows counts for the number of essays submitted. Although the values for this feature are numeric, there is no meaningful interpretation for allowing fractions of an essay, and so I treat `essays` as a discrete feature. With the exception of 0, the other levels have sufficient counts to ensure reliable proportion estimates. As with `birthyear`, the single occurrence of 0 will be handled by Laplace smoothing.

Table 3: Counts for Number of Essays Submitted

Number of essays	0	3	4	5
Count	1	63	58	19

2.2 Verbal and math SAT scores

Figure 1 shows histograms for the verbal and math SAT scores. Overlain in blue are the respective normal densities with parameters

$$\begin{aligned} \mu &= 674.61, \quad \sigma = 81.91 \quad \text{for } \texttt{verbal}, \text{ and} \\ \mu &= 674.47, \quad \sigma = 75.70 \quad \text{for } \texttt{math}, \end{aligned}$$

obtained by computing the sample mean and standard deviation across all instances for each feature. Because the normal densities fit the underlying histograms reasonably well, I will model both SAT scores using normal distributions with the above parameters.

2.3 Cumulative GPA

Figure 2 shows a histogram for the cumulative GPAs in the data set. Overlain in blue is a normal density curve with parameters $\mu = 3.386$ and $\sigma = 0.347$, estimated from the observed GPAs. The curve does not fit the histogram very well; in particular, it overestimates the sample mean. As such, I also try out the other pre-processing method by discretizing the GPAs into low ($\texttt{gpa} \leq 3.2$), medium ($3.2 < \texttt{gpa} \leq 3.5$), and high ($\texttt{gpa} > 3.5$). The red dotted lines in Figure 2 indicate these break points, and show that there are sufficient counts in each category for accurate proportion estimates. I compare the accuracies obtained by both pre-processing methods in §4.

Figure 1: Histograms of Verbal and Math SAT Scores

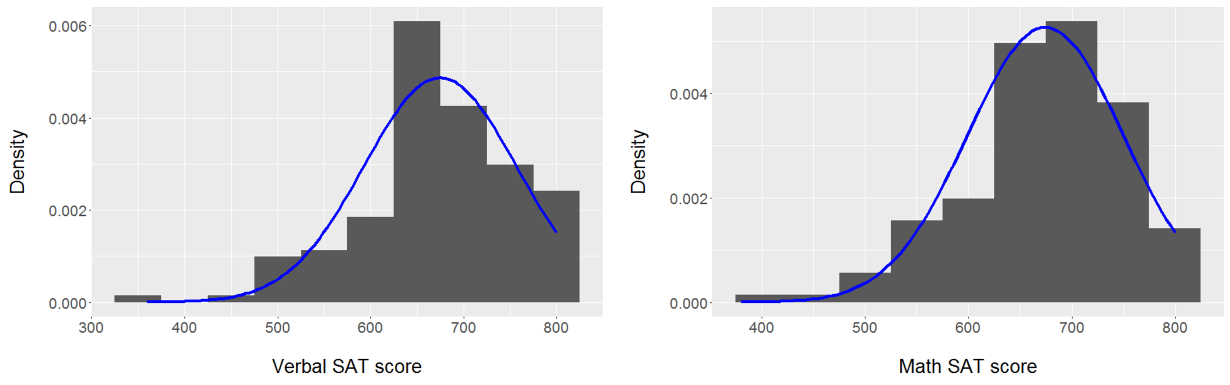
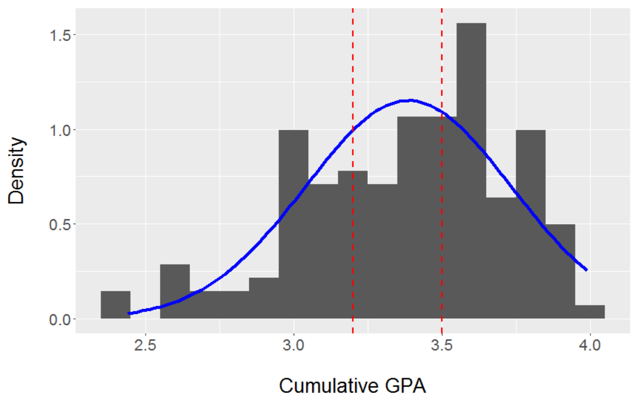


Figure 2: Histogram for Cumulative GPA



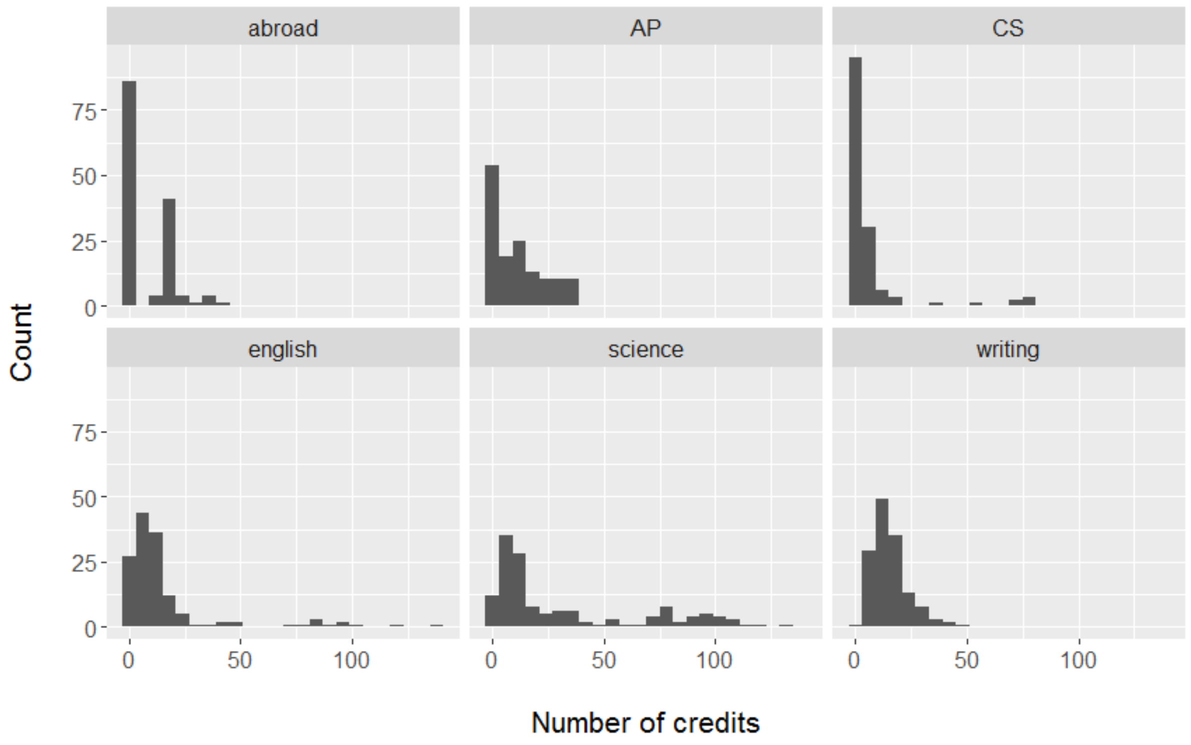
2.4 Number of credits

Figure 3 shows histograms for various types of credits. Of the six panels, only the lower right for `writing` seems to follow a distribution, which means that I have to discretize the data for all the other five features. For `abroad`, `AP` and `CS`, I classify the feature values into three factor levels: low, medium and high. To account for the larger spread in values for `english` and `science`, as reflected in the histograms, I use a fourth level: very high. Table 4 shows my chosen break points, designed to ensure sufficient counts for each level.

Table 4: Break Points for Discretized Number of Credits

Feature	Low	Medium	High	Very high
`abroad`	≤ 10	$10 < . \leq 16$	> 16	-
`AP`	$= 0$	$0 < . \leq 24$	> 24	-
`CS`	$= 0$	$0 < . \leq 18$	> 18	-
`english`	≤ 3	$3 < . \leq 6$	$6 < . \leq 36$	> 36
`science`	≤ 6	$6 < . \leq 36$	$36 < . \leq 80$	> 80

Figure 3: Histograms for Various Types of Credits



2.5 Number of writing credits

Figure 3 showed that the number of writing credits may follow a known probability distribution. I inspect this more closely by means of Figure 4, which replicates the histogram for `writing`. The blue dots show the height of the probability mass function of a negative binomial distribution with parameters $r = 4.675$ and $p = 0.232$, estimated from the data. Although the fit is almost exact, the negative binomial distribution is less frequently called upon to model data involving counts, and so I also try out discretization to see if using the negative binomial distribution leads to significant accuracy gains. I bin the values for `writing` into low ($\texttt{writing} \leq 10$), medium ($10 < \texttt{writing} \leq 20$), and high ($\texttt{writing} > 20$). The red lines in Figure 4 indicate that the chosen break points partition the data fairly well into evenly-sized subsets.

3 Feature Selection

Not all of the features have to be used in the classification. In particular, whether or not a student lives in Minnesota and the student's birth year should have no effect on whether he or she passes the writing portfolio. Figure 5 gives evidence for this by displaying the relative proportion of students who were given a "needs work", grouped by the factor levels of `mn` and `birthyear`. With the exception of birth years 1980 and 1982, the proportion of "needs work" versus otherwise is roughly 50-50 across all factor levels. This suggests that the proportion of "needs work" does not seem to depend much on the values of these two features. I remove each of these two features in turn and report the effect of omitting either on the classifier's accuracy.

Figure 4: Histogram for Number of Writing Credits

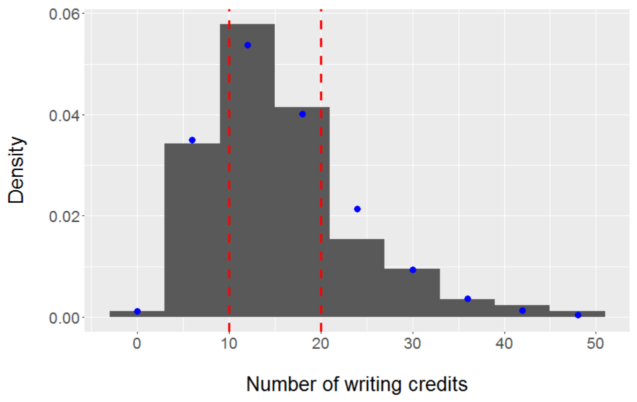
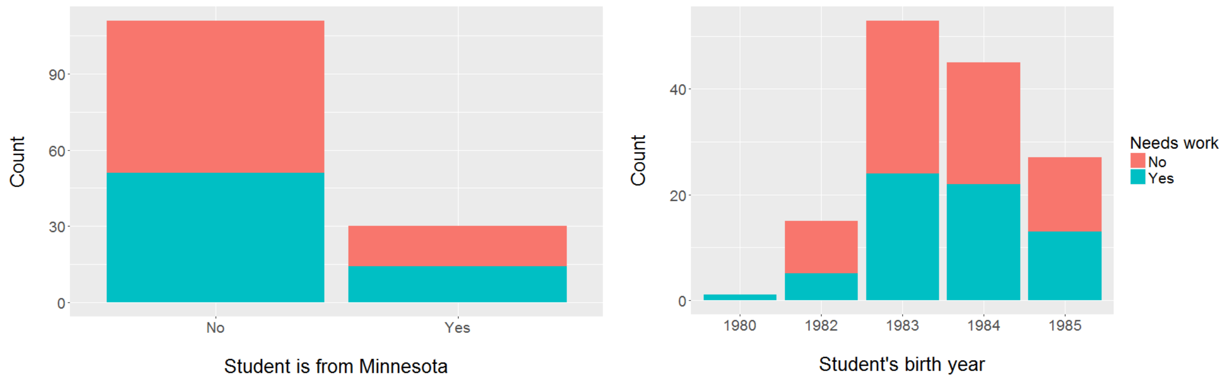


Figure 5: Relative Proportion of Needs Work, Grouped by Factor Levels of `mn` and `birthyear`



4 Results and Discussion

Table 5 shows the leave-one-out cross validation accuracy rates for variations of my Naive Bayes classifier as described in §2 and §3. All variations use a default Laplace smoothing parameter of $\alpha = 1$. Discretizing `gpa` and `writing` both led to accuracy gains, which indicates that the normal distribution for the former and negative binomial distribution for the latter were not worth using. In fact, discretizing `writing` led to the single biggest gain in accuracy, suggesting that the negative binomial density somehow failed to model the data despite being a nearly exact fit. Removing `mn` had no effect on the accuracy rate, which means that it can be left out at no expense but with the benefit of reduced dimensionality. Lastly, omitting `birthyear` causes accuracy to fall but not by much, indicating that `birthyear` contains marginally useful information for the classification.

Table 5: Leave-one-out Cross Validation Accuracy Rates for Variations of the Classifier

Variation	baseline	discrete `gpa`	discrete `writing`	no `mn`	no `birthyear`
Accuracy rate	74.47%	75.18%	78.01%	74.47%	73.76%

Based on the results in Table 5, I select my "best" classifier, which discretizes both `gpa` and `writing`, and omits only `mn`. The leave-one-out cross validation accuracy for this "best" classifier is 79.43%, which indicates that the gains from the modifications above are somewhat cumulative. Although not shown here, increasing α too far beyond 1 led to significant decreases in accuracy, while decreasing α below 1 did not change the accuracy at all.

The confusion matrix for my "best" classifier is shown in Table 6. The False Positive Rate (FPR) is $13/(13+63) = 17.1\%$, while the False Negative Rate (FNR) is $16/(49+16) = 24.6\%$. Since the FNR is higher, my classifier is over-predicting the number of not "needs work", and so I focus on students who did not pass the writing portfolio but whom the classifier failed to classify as "needs work".

Table 6: Confusion Matrix for "Best" Classifier

		Predicted	
		needs work	not needs work
Actual	needs work	49	16
	not needs work	13	63

By looking at the distribution of each feature for the false negative cases (not shown here), I found that verbal SAT scores tended to be near the mean for the data set, while math SAT scores tended to be high. This shows that the classifier has over-associated mean verbal scores and high math scores with the student's portfolio not needing work. The vast majority of portfolios that were false negatives also only had three essays, indicating that the classifier has over-associated fewer essays with a higher likelihood of passing. Lastly, for the number of credits besides those for AP and writing, the false negative cases tended to have low credits. But this occurs not because of a failing of the classifier, but simply because a large proportion of students in the data set had low numbers of credits to begin with. The distributions for the other unmentioned features are fairly uniform, and so these features are unlikely to be the main contributors to the false negatives.

The Naive Bayes classifier uses the assumption of conditional independence among all features given the class label. I test whether pairwise conditional independence holds for this data set by estimating the conditional joint distributions for two discrete features given a class label. First, I look at `mn` and `int`. Table 7 gives their conditional joint distribution given a class label of "needs work". The estimated probabilities are obtained by normalizing the counts.

Table 7: Conditional Joint Distribution for `mn` and `int` Given "needs work"

`mn`	`int`	Count	Estimated probability
0	0	46	0.708
0	1	5	0.0768
1	0	14	0.215
1	1	0	0

Based on Table 7, the product of the marginal conditional probabilities for $\texttt{mn} = 0$ and $\texttt{int} = 0$ is

$$P(\texttt{mn} = 0|\text{needs work})P(\texttt{int} = 0|\text{needs work}) = (0.708 + 0.0768)(0.708 + 0.215) = 0.724.$$

This is different from the joint conditional probability $P(\texttt{mn} == 0, \texttt{int} == 0|\text{needs work}) = 0.708$, and so in general

$$P(\texttt{mn}, \texttt{int}|\text{label}) \neq P(\texttt{mn}|\text{label})P(\texttt{int}|\text{label})$$

Because the difference between the two computed numbers above could be the result of noise, I try out another pair `int` and `essays`. Table 8 gives their conditional joint distribution given a class label of "needs work". Notice that

$$\begin{aligned} P(\texttt{mn} = 1|\text{needs work})P(\texttt{essays} = 4|\text{needs work}) &= (0.0462 + 0.0308)(0.277 + 0.0308) \\ &= 0.0237 \\ &\neq 0.0308 \\ &= P(\texttt{mn} = 1, \texttt{essays} = 4|\text{needs work}) \end{aligned}$$

Table 8: Conditional Joint Distribution for `int` and `essays` Given "needs work"

`mn`	`essays`	Count	Estimated probability
0	0	1	0.0154
0	3	33	0.508
0	4	18	0.277
0	5	8	0.123
1	0	0	0
1	3	3	0.0462
1	4	2	0.0308
1	5	0	0

I have shown that pairwise conditional independence fails to hold for $\{\texttt{mn}, \texttt{int}\}$ and $\{\texttt{int}, \texttt{essays}\}$. It follows that conditional independence fails to hold in general for this data set.

5 Comparison With k-Nearest Neighbors

Comparing the runtime for this assignment to that from Homework #1, it is obvious that the Naive Bayes (NB) algorithm is much faster than the k-nn algorithm. This could be due to different data set sizes, but the NB implemented here uses leave-one-out cross validation, which is much more computationally intensive than the 5-fold cross validation used for the k-nn algorithm in Homework #1.

NB is also much easier to implement compared to the k-nn.

The major disadvantage of NB is its assumption of conditional independence among its features. As shown in §4, this assumption does not always hold for data sets. This is unlike the k-nn, which makes no assumption about independence among features.

Another con of NB is that its performance is rather sensitive to whether certain features are treated as discrete or continuous, and to the type of distribution used to model continuous features. This is unlike the k-nn, which does not try to model how the feature values are distributed.

NB and k-nn are both lazy learning algorithms, in the sense that no training of any weights is necessary before the classifier can be used on the test set. The only "training" required is to store the training data and in the case of NB, computing conditional probabilities beforehand.

NB and k-nn are different, however, because the former uses probabilistic structures within the data set to classify test instances, whereas the latter uses a brute-force method of finding the closest neighbors for each test instance. This accounts for why k-nn is much slower compared to NB.

I would use k-nn over NB in the following cases:

- the data set has features for which I cannot make any distributional assumptions about (e.g. normality, discrete or continuous)
- the data set has correlated features, which would cause NB to classify wrongly more often
- I expect the decision boundary to be non-linear or non-parabolic, which NB is incapable of creating

I would use NB over k-nn in the following cases:

- I have a large test set, which would make k-nn very slow
- I want to get a sense of which features are contributing the most information to the classification, which k-nn cannot tell me
- the data set has a lot of missing data, which would cause k-nn to compute undefined distances; NB, on the other hand, handles this well due to Laplace smoothing