ASSIGNMENT - 2 (Part 2)

MACHINE LEARNING (BITS F464)

Prithvi Raj - 2016A7PS0013P
Adit Chandra - 2016A3PS0256P
Dhaivat Pandya - 2016A7PSP0020P

PROBLEM STATEMENT

To come up with questions and answers to those questions based on the given data of marks of students in a course. Also build some models to predict the final grades based on midsem and other attributes.

ABSTRACT

After analysing the data, we have answered the following question.

1. Can we predict the final grades of the students using the given data ?
2. Which attribute contributes the most to the final grade ?
3. Is PCA a good choice for dimensionality reduction ?
4. Is there any causality relationship between the data ?

We also completely made and tested two models to predict the final grade based on the midsem performance and other attributes available in the data. We madethe following models and analysed their performance.

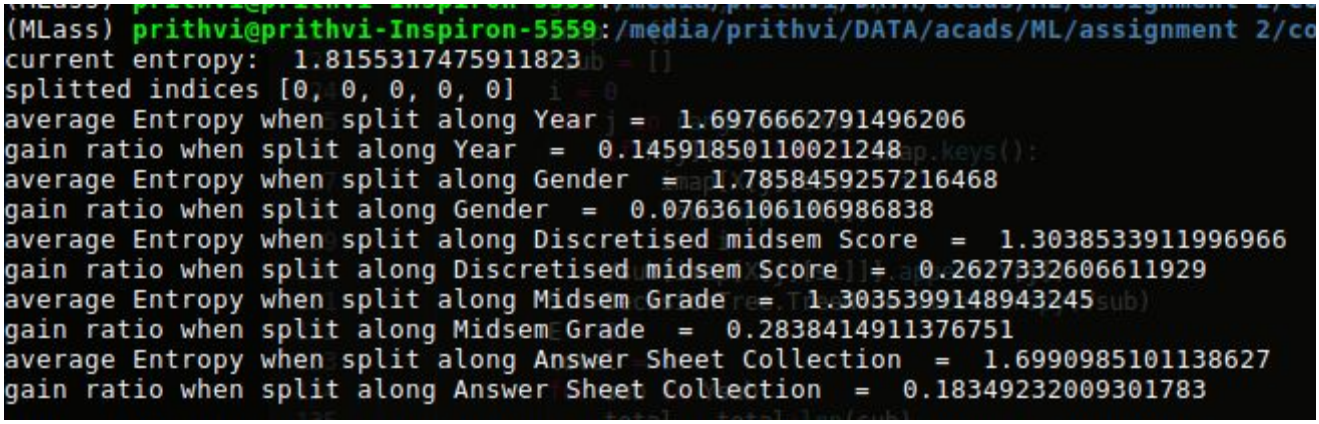
1. Decision tree classifier
2. Naive bayes classifier

INFERENCES FROM THE DATA

1. Can we predict the final grades of the students using the given data ?
Yes, prediction of final grade would be a classification problem in which the class is the grade. There are 8 classes corresponding to each of the grades.

2. Which attribute contributes the most to the final grade ?
We decided that it was best to answer this question using the concept of decision entropy (discussed in detail in the section on decision tree classifier). The idea is that we split the data along each possible attribute and see which attribute produces the most biased sets with regards to the output variable (output variable here is the end sem grade). This notion of biased sets is captures well by the concept of decision entropy. If an attribute has little contribution to the final output, then the information we gain out of it is also less (this is why decision trees are good at handling irrelevant attributes)

The following image is a screen shot from the output shown during the decision tree learning process. The output shows the process of finding the best attribute to split along at the root of the decision tree (i.e when the entire data set is available). We calculate the entropy for the subsets formed when we split along each attribute. Whichever attribute split produces the least entropy (or highest information gain) has the most effect in deciding the final class of the input.



As you can see, the least entropy is produced by splitting along midsem grade. Which means midsem grade has the highest contribution towards deciding the final grade.
From a student's perspective, we can agree that this answer indeed makes sense because as students we know it is quite likely your midsem and final grades would be the similar.

3. Is PCA a good choice for dimensionality reduction ?

We feel PCA is not a good choice regardless of the context of the problem. The only reason one would use PCA is there is some severe lack of computing power. Otherwise it makes no sense to use PCA because the side effects of PCA grossly outweigh any of it's benefits.

Pros of PCA

1. Decreases model complexity allowing for faster learning.

Cons of PCA

1. PCA does blind reduction of dimensions only based on variance without looking at the class labels. So it is very likely that some important attribute which would have allowed for classification gets removed. So if there is an irrelevant attribute with high variance, it will be chosen over a relevant attribute with low variance
   Eg. say some attribute which has only two values, but those two values split the data quite evenly into two classes, this attribute will have very low variance and will most likely be neglected by PCA
2. Tends to find linear correlations in data.
   Eg. say two of the attributes are length in inches and length in centimeters. Then after normalisation (making mean zero and dividing by the range (not standard deviation)) these two attributes would have the same variance. So if the variance is high, then both these attributes might end up in our final set.

4. Is there any causality relationship between the data ?

Being students (domain experts), we are confident that there is significant positive correlation between CGPA and attributes like midsem grade and final grade.
We wrote some code to find the covariance between CGPA and midsem grade and also between CGPA and end sem grade and the values showed positive correlation.

DECISION TREE CLASSIFIER

Decision tree just decides at each node of the tree, which branch to take to go down based onThe value of an attribute. Once we reach a leaf node, the estimation for the class of the input can be made. The learning process for a decision tree is just a matter of picking the right attribute to split along at each node. This is where the concept of entropy comes in.

Entropy: Entropy is a measure of how biased a given set is with respect to the classes of the classification problem. Given a set **S** of labeled inputs in a k class classification problem, the entropy for that set is calculated as

$$E(S) = -\sum_{all\ j} P(j \mid S) \,.\, log\,(P(j \mid S))$$

Where j represents a class among the k classes and $P(j \mid S)$ is the probability of that class occuring in the set S.

The information gain when you split a set S along an attribute A is the weighted average of the entropies of all the subsets S_v for all values v of A.

$$Gains\,(S,\ A) = E(S) - \sum_{all\ v \in A} \frac{|Sv|}{|S|} E(S_v)$$

At each node we can split along the attribute which produces the highest information gain.

But this has the disadvantage that the number of sets produces might be very large so the so the entropy will be high because each of those small sets might be pure. At higher levels of the tree, we want to do splits that generate as large groups of biased sets as possible. We do this by incorporating split entropy.

$$Split\ Entropy\,(S,A) = -\sum_{all\ v \in A} \frac{|Sv|}{|S|} log(\frac{|Sv|}{|S|})$$

We can see that split entropy will be high is a split produces a large number of small sets. Therefore we want to minimise the split entropy. So we define gain ratio as the gain divided by the split entropy of a split.

$$Gain\ ratio\,(S,A) = \frac{Gaind\,(S,A)}{Split\ Entropy\,(S,A)}$$

We finally split along the that attribute which has the higheset gain ratio.

Learning algorithm: Recursively split the data along the best attribute until one of the following become true.

1. Set only has points in belonging to one class. At this node, the predicted class would obviously be that class
2. All attributes have already been split along. At this node, the predicted output is the most likely class in the subset (i.e. the class which occurs the most)

3. Set is smaller than the minimum set (minimum size set to prevent overfitting). At this node, the predicted output is the most likely class in the subset (i.e. the class which occurs the most)

Implementation Details:

- We used python to implement the decision tree.
- We had to remove the attribute CGPA because there were too many missing values and decision trees are not good at handling missing values.
- We discretised the midsem score by bucketing (dividing by 5) and also rounded up the CGPA. This was done to ease the implementation as the decision tree for continuous values is much more difficult to implement.

NAIVE BAYES CLASSIFIER

In machine learning, **naive Bayes classifiers** are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naive) independence assumptions between the features.
Naive Bayes classifiers are highly scalable, requiring a number of parameters linear in the number of variables (features/predictors) in a learning problem. Hence, maximum-likelihood training can be done by evaluating a closed-form expression, which takes linear time, rather than by expensive iterative approximation as used for many other types of classifiers. With appropriate preprocessing, such classifiers are competitive with more advanced techniques such as support vector machines.

Probabilistic Model - Abstractly, naive Bayes is a conditional probability model: given a problem instance to be classified, represented by a vector $\boldsymbol{x} = (x_1,\ldots,x_n)$ representing some n features (independent variables), it assigns to this instance probabilities

$$p(C_k \mid x_1, \ldots, x_n)$$

for each of K possible outcomes or *classes* C_k.

The problem with the above formulation is that if the number of features n is large or if a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable. Using Bayes' theorem, the conditional probability can be decomposed as

$$p(C_k \mid \mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x} \mid C_k)}{p(\mathbf{x})}$$

The numerator is equivalent to the joint probability mode which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability

$$\begin{aligned} p(C_k, x_1, \ldots, x_n) &= p(x_1, \ldots, x_n, C_k) \\ &= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2, \ldots, x_n, C_k) \\ &= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) p(x_3, \ldots, x_n, C_k) \\ &= \ldots \\ &= p(x_1 \mid x_2, \ldots, x_n, C_k) p(x_2 \mid x_3, \ldots, x_n, C_k) \ldots p(x_{n-1} \mid x_n, C_k) p(x_n \mid C_k) p(C_k) \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature x_i is conditionally independent of every other feature x_j for $j \neq i$, given the category C_k.

$$p(x_i \mid x_{i+1}, \ldots, x_n, C_k) = p(x_i \mid C_k)$$

Thus, the joint model can be expressed as

$$\begin{aligned} p(C_k \mid x_1, \ldots, x_n) &\propto p(C_k, x_1, \ldots, x_n) = \\ &= p(C_k)\ p(x_1 \mid C_k)\ p(x_2 \mid C_k)\ p(x_3 \mid C_k)\ \cdots \\ &= p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)\,. \end{aligned}$$

Classifier - The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function that assigns a class label $y = C_k$ for some k as follows:

$$\hat{y} = \underset{k \in \{1, \ldots, K\}}{\operatorname{argmax}}\ p(C_k) \prod_{i=1}^{n} p(x_i \mid C_k)$$

Gaussian naive Bayes - When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution. For example, suppose the training data contains a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_k be the mean of the values in x associated with class C_k, and let σ_k be the standard deviation of the values in associated with class C_k. Suppose we have collected some observation value v. Then,

$$p(x = v \mid C_k) = \frac{1}{\sqrt{2\pi\sigma_k^2}}\, e^{-\frac{(v-\mu_k)^2}{2\sigma_k^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features; some literature in fact suggests that this is necessary to apply naive Bayes, but it is not, and the discretization may throw away discriminative information.

Implementation Details - The naive Bayes classifier built for the given data has input vectors of length 6 (consisting of attendance, gender, CGPA, midsem marks, midsem grade, collection attempt) and output is a single scalar, corresponding to the grade point given for the grade. The classifier has two continuous variables, CGPA and midsem marks, and a gaussian distribution is assumed for them. The other variables are considered to be class labels, and frequentist probabilities are used as priors.

COMPARISON

Since the data given to us was limited we had to come up with a way to run multiple different tests. We decided split the data as 80% training and 20% test data after randomly sorting the it. This means the training and test data generated each time would be slightly different.
The following table shows the results.

	Midsem	NBC	Decision Tree
Test 1	1.35897435897436	1.28205128205128	1.58974358974359
Test 2	1.35897435897436	1.07692307692308	1
Test 3	1.38461538461538	1.35897435897436	1.35897435897436
Test 4	1.28205128205128	1.07692307692308	1.25641025641026
Test 5	1.1025641025641	1.1025641025641	1.35897435897436

The error function we used is just average of the absolute difference between predicted grade point and actual grade point.
The second column shows “midsem prediction” i.e. predicting the final grade as the same as midsem grade.

As you can see, due to lack of data, the learning process is not very effective and therefore the errors are quite unpredictable.