CSE 601:DATA MINING and BIOINFORMATICS

Classification Algorithms

Ashutosh Ahmad Alexander	50248859	alexanda@bufalo.edu
Dilip Reddy Gaddam	50248867	dilipred@buffalo.edu
Pradeep Singh Bisht	50247429	pbisht2@buffalo.edu

K-Nearest Neighbors:

1. K- Nearest Neighbors is classification algorithm in which an object is assigned to class or label based on its k-nearest neighbors. The object is assigned to a class which is most common among its K neighbors.

2. K-NN is called lazy learning Algorithm because all computation is deferred until classification.

Implementation of K-NN:

We have Implemented Naïve Bayes classifier in Python. Our Program performs 10-fold cross validation by choosing different training data and testing data in every iteration and outputs the average accuracy, precision, recall and F-Measure after the 10 iterations.

1. Choose the value of K, Where K is the number of nearest neighbors of unclassified label to compare and assign.
2. Read the data from the file and divide it into training and test datasets.
3. Take an unclassified data set from testing data and calculate the Euclidean distance from all the points in training data to this point.
4. Pick top K closest neighbors and check the labels of neighbors.
5. Assign the label which occurs in majority to our training dataset.
6. Repeat the same process for all the values in test data.
7. calculate the performance of model by comparing the assigned to original label of test data.
1. The same process is iterated for 10 iterations and average accuracy is calculated as average of accuracies in all iterations.

In K-NN Algorithm choosing K value is important because:

1. If K value is too low algorithm becomes sensitive to noise and it has very less information to classify the algorithm.
2. If K value is too large neighborhood may include points from other classes. i.e. Large amount of un-necessary information is given to algorithm.

Results:

For project3_dataset1.txt:

K Value	Average Accuracy	Average Precision	Average Recall	Average F-Measure
3	96.64%	98.06%	92.26%	0.950
5	96.34%	96.67%	92.83%	0.946
10	97.19%	98.90%	93.06%	0.96
15	97.19%	98.91%	93.06%	0.958

20	95.97%	98.858%	89.6%	0.93

For project3_dataset2.txt

K Value	Average Accuracy	Average Precision	Average Recall	Average F-Measure
3	64.77%	53.482%	48.69%	0.506
5	63.69%	51.76%	43.14%	0.46
10	69.29%	63.09%	42.99%	0.508
15	68.53%	62.73%	40.23%	0.488
20	68.65%	65.65%	41.01%	0.502

Result Analysis:

1. K-NN provides very accuracy on dataset1 when compared to dataset2 because dataset1 contains only continuous data but dataset2 contains both continuous and Categorical data. This implies that K-NN does not perform well on categorical data or the data which contains both categorical and continuous data.

2 Results show that performance of algorithm is low when the k value is too low or too high. So k value should never be too high or too low.

Pros:

1. K-NN is easy to implement and robust to noise training data.
2. It works well on large datasets.

Cons:

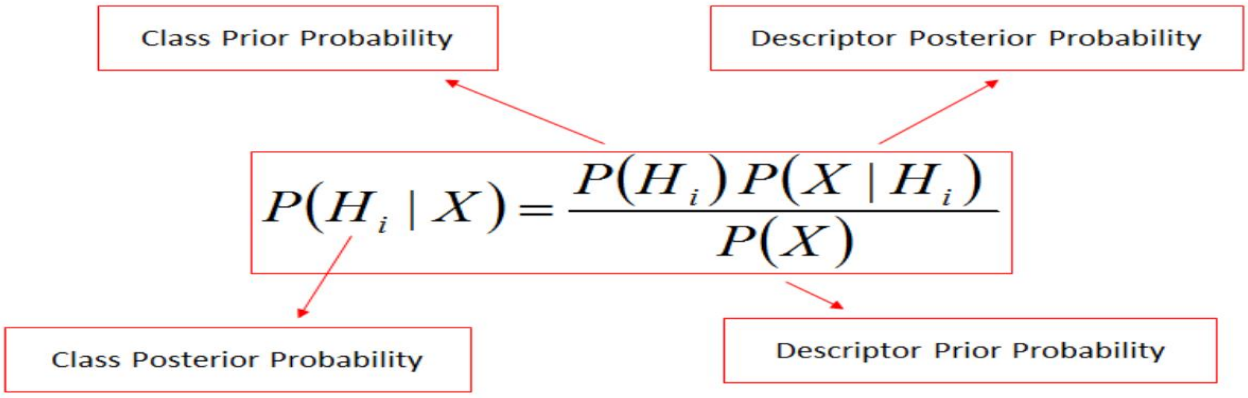
1. Choosing the value K is difficult.
2. Normalization of data needs to be carried out to avoid domination of distance measures by any attributes.

Naive Bayes Classification:

1. Naive Bayes is a probabilistic classification algorithm based on Bayes' Theorem. Naïve Bayes calculates the posterior probability in order to classify the given data.
2. Naïve Bayes assume all the attributes are independent and all of the given features have equal probability of occurrence.
3. Naïve Bayes works efficiently on large datasets and is also easy to implement.

Bayes Theorem:

Bayes Theorem calculates the Class Posterior probabilities of given data with independent variables and classify the data into the class with highest Posterior probability.



Implementation of Naïve Bayes:

We have Implemented Naïve Bayes classifier in Python. Our Program performs 10-fold cross validation by choosing different training data and testing data in every iteration and outputs the average accuracy, precision, recall and F-Measure after the 10 iterations.

Steps Followed:

1. Read the input data from the file.
2. Identify all columns with numerical value and categorical values and save the indices in two different lists.
3. Using the 10-fold cross validation split the data into training and testing data with test data changing in every iteration.
4. Calculate the class prior probabilities of each class in the training data.
5. Calculate the Mean and Standard deviation of numerical attributes in the training data for each class.
6. Calculate the Descriptor posterior probability for numerical attributes of the given test data for each class using probability density function, mean and standard deviation calculate din the previous step.

$$P(x_i|y) = \frac{1}{\sqrt{2\pi\sigma_y^2}} exp\left(-\frac{(x_i-\mu_y)^2}{2\sigma_y^2}\right)$$

7. Calculate the Descriptor posterior probability for Categorical attributes of the given dataset. If there are no categorical attributes in the data, then probability is considered as One.
8. Calculate the Posterior probabilities of each class by multiplying class prior probability and descriptor prior probability of each class.

9. Classify the data to that class which has highest class posterior probability.
10. Calculate accuracy of the classification by comparing the obtained results with original class of the test data.

Results:

1. project3_dataset1.txt:

Average Accuracy: 93.5%	Average Precision: 92.4%
Average Recall: 90.54%	Average F-Measure: 0.913

2. project3_dataset2.txt:

Average Accuracy: 70.2%	Average Precision: 57.1%
Average Recall: 61.7%	Average F-Measure: 0.584

Result Analysis:

1. By comparing the average results of our model on two datasets we can say that our model performed well on dataset1 when compared with dataset2 which has the accuracy of 70.2%. The reason what we think for the decrease in accuracy of the model is, dataset1 contains only continues values whereas dataset2 contains both continuous and categorical values. This could have been a reason for decrease in accuracy of the model.

2. One more important thing that could be the reason for decrease in accuracy of the model is consideration of independence among all the attributes of the data. There might be some correlated attributes in dataset2 which could have resulted in decrease in accuracy.

Pros:

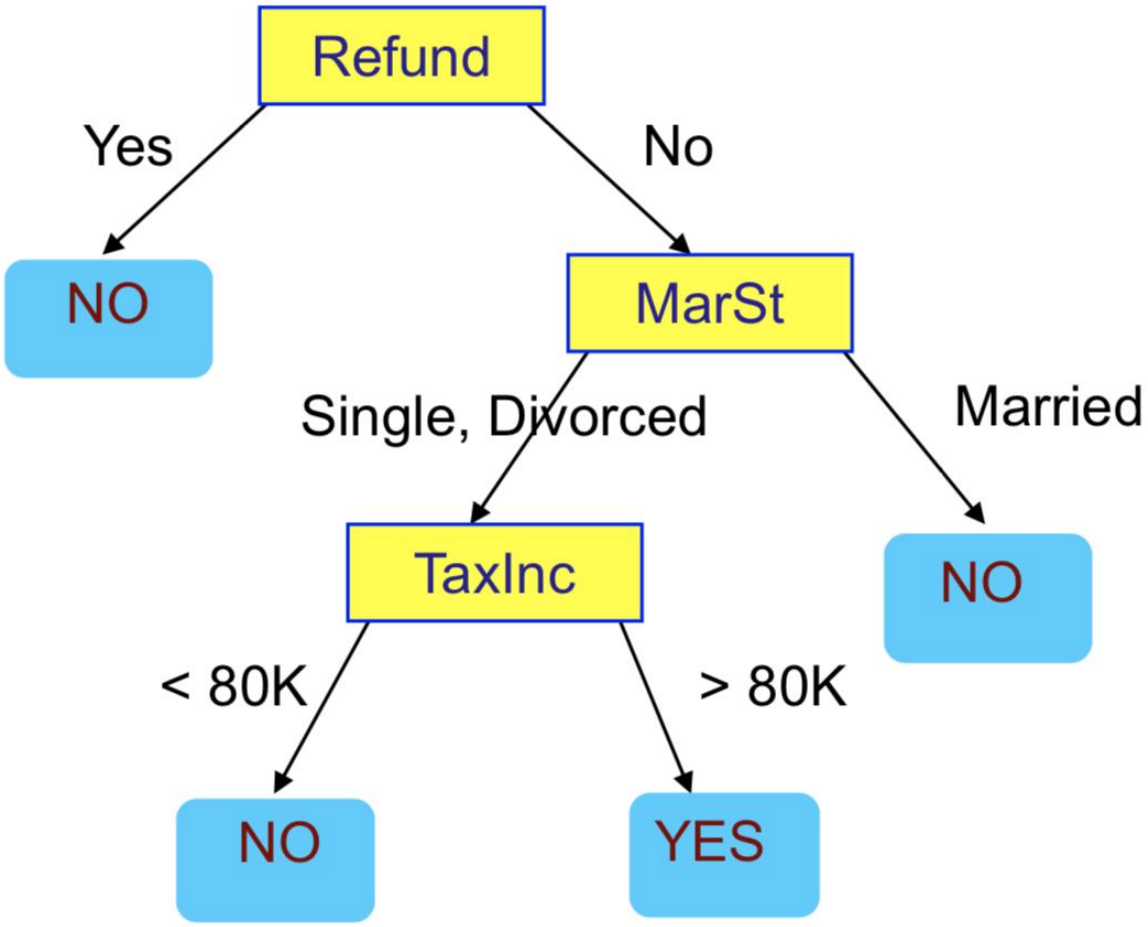
1. Easy to implement and works well on large datasets.
2. Naive Bayes classifier has the performance which is comparable to that of Decision trees.
3. Works on both continuous and categorical data.

Cons:

1. Naïve Bayes assumes all the attributes are independent and doesn't work well on datasets with correlated data.
2. Naïve Bayes fails when the test data has new labels that were not present in the training data.

Decision Tree.

A decision tree is a graph that uses a branching method to illustrate every possible outcome of a decision. Each internal node represents a ‘test’ on an attribute, each branch represents the outcome of the test, and each leaf node represents a class label. The paths from root to leaf represent classification rules.



Implementation details:

1. The entire dataset is split into training and testing sets using 10-fold validation.
2. Training set is used to train the model to construct a decision tree.
3. Records based on all values of a column are taken. (Each column represents an attribute).
4. The algorithm is performed on the column and value which provides minimum GINI score is taken as it provides the maximum gain.
5. The above steps are recursively called for both left and right child of the node.
6. At any stage, if all the records belong to the same class, then it is made a leaf node with a particular label for that class.

7. This algorithm is repeatedly called for each subset until it cannot be split further.
8. Finally, the accuracy from each iteration is taken to find the average accuracy for the whole dataset.

Result:

Dataset	Avg Accuracy	Avg Precision	Avg Recall	Avg F-Measure
project3_dataset1.txt	91.9204260652	89.1746693005	89.0266545049	0.888518106923
project3_dataset2.txt	59.3061979648	41.105993207	45.0542177121	0.423173921884

Pros:

1. Are simple to understand and interpret.
2. Can handle both numerical and categorical data. Can also handle multi-output problems.
3. It implicitly performs feature selection.
4. Nonlinear relationships between parameters do not affect tree performance.

Cons:

1. It can result in overfitting.
2. Can be unstable because small variations in the data might result in a completely different tree being generated.
3. Can create biased trees if some values dominate.

Boosting

Boosting is a machine learning ensemble algorithm primarily reducing bias and variance. This helps in converting weak learners to strong learners.

Implementation details:

For our project, we have used **AdaBoost** or Adaptive Boosting which uses the following technique:

1. The entire dataset is split into training and testing sets using 10-fold validation.
2. Training set is used to train the model to construct a decision tree.
3. Choose L – number of classifiers.
4. Set initial weights to all the training data as 1/size of training data.
5. Use weighted bagging to train the training set.

6. Each classifier is trained on data, taking into consideration the previous classifier's success.

7. Calculate error by adding all the weights where items have been misclassified divided by sum of all weights.
8. After training each step of training process, a weight is assigned to each sample in the training set is set equal to the current error on the sample.
9. These weights can be used to inform the training of the weak learner, and subsequent learners will focus on those during their training.
10. Test the test dataset by predicting the label of every tree out of L.
11. Multiply the label with the corresponding alpha for that particular tree such that if it is predicted 1, then it should be 1*alpha else -1*alpha if predicted 0.
12. Finally, the accuracy from each iteration is taken to find the average accuracy for the whole dataset.

Result:

For Dataset1: project3_dataset1.txt

Number of Learners	Avg Accuracy	Avg Precision	Avg Recall	Avg F-Measure
5	95.078320802	93.7406546302	92.8036552819	0.9320856204
10	95.078320802	93.5508750509	93.1129522651	0.932235306552
15	96.484962406	97.1662274923	93.5166548427	0.952176340159

For Dataset2: project3_dataset2.txt

Number of Learners	Avg Accuracy	Avg Precision	Avg Recall	Avg F-Measure
5	62.5578168363	46.0144300144	39.4898113582	0.409119099011
10	62.3404255319	44.5661054632	38.5690566743	0.40524212765
15	64.2784458834	48.1438365556	44.8153922101	0.457385149966

Pros:

1. Less susceptible to the overfitting problem to other learning algorithms.

Cons:

2. Sensitive to noisy data and outliers.

Random Forest:

Random decision forests are an ensemble learning method for classification, regression, and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes.

Bagging: It is a machine-learning ensemble technique designed to improve the stability and accuracy of algorithms used in classification. It also reduces variance and helps to avoid overfitting.

Implementation details:

1. The entire dataset is split into training and testing sets using 10-fold validation.
2. Training set is used to train the model to construct a decision tree.
3. Choose T – number of trees in each node.
4. We use **bagging** where we choose a training set N times with replacement from the training set.
5. For each node, we randomly select m features (m < M, where M is total number of features) and calculate the best split.
6. The we take the majority voting among all the trees to decide a label for a record.
7. Finally, the accuracy from each iteration is taken to find the average accuracy for the whole dataset.

Result:

For Dataset1: project3_dataset1.txt

Number of Trees	Avg Accuracy	Avg Precision	Avg Recall	Avg F-Measure
5	94.727443609	91.312647526	94.8767947029	0.930027964433
10	95.0845864662	96.280769976	90.752812405	0.932859363328
15	95.9586466165	96.2460254721	96.2460254721	0.945749647645

For Dataset2: project3_dataset2.txt

Number of Trees	Avg Accuracy	Avg Precision	Avg Recall	Avg F-Measure
5	66.0175763182	50.5681723248	44.0175234123	0.465934065934
10	66.4754856614	52.007020757	45.911240806	0.473168532123
15	66.2580943571	52.143018746	40.8950552372	0.443636247085

Pros:

1. Reduction in overfitting: by averaging several trees, there is a significantly lower risk of overfitting.

2. Less variance: By using multiple trees, you reduce the chance of stumbling across a classifier that doesn't perform well because of the relationship between the train and test data.

Cons:

1. More complex compared to decision trees.
2. Hard to visualize the model.
3. More computationally expensive.