CSE 601: DATAMINING AND BIOINFORMATICS

Project 3 : Classification Algorithms

By:
Naina Nigam(50208030)
Surabhi Singh(50208675)
Vanshika Nigam(50208031)

K Nearest Neighbors

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (for example, distance). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

- **Algorithm:** A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbor. Following distance functions are used for calculating distance between numerical data points - Euclidean, Manhattan, Minkowski. These are valid only for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

- **Implementation:** Following are the implementation steps followed in this project-

  Step 1- Dataset file (project3_dataset1.txt or project3_dataset2.txt) and checked for presence of categorical data in the columns. If any column has categorical data then it is encoded by counting unique values in the column and putting those in a set. The index of the set for those values is then used as the encoded values for original categorical data.

  Step 2- 10 fold cross validation is used to split data into training dataset and test dataset by first calculating the size of each fold. Split is done by picking each fold sequentially. Before splitting the dataset, it has been shuffled.

  Step 3- After splitting, first training dataset is normalized using its mean and standard deviation values then same mean and standard deviation values are used for normalizing test dataset. After normalizing, both train and test datasets are passed to k_nearest_neighbor method.

  Step 4- In this method, test data is iterated and every row of test data along with the train dataset is sent to the distance_cal method where distance between rows is calculated. So for every test data row there is a list of distances and the index of corresponding row of train data. For continuous values euclidean distance is used and for the encoded categorical values hamming distance is used.

Step 5- The list of distances is sorted and then top k smallest (k nearest) values are computed. Then the label with maximum frequency among the k neighbors is chosen and assigned to that particular test data row.

Step 6- After labels for the whole test data have been computed, these are matched against the original test data labels and following metrics : accuracy, recall, precision, F-measure are calculated for each fold.

Step 7- At last the average for all metrics is computed for the 10 folds and displayed as the result.

Formula used to calculate the 4 performance metrics:

- ❖ Accuracy: (true negative + true positive)/(true negative + true positive + false negative + false positive)
- ❖ Precision: true positive/(true positive + false positive)
- ❖ Recall: true positive/(true positive + false negative)
- ❖ F measure: (2*recall*precision)/(recall + precision)

- **Results:**

Results for project3_dataset1.txt: For k=5

- ❖ accuracy= 0.9683897243107771(~96.84%)
- ❖ recall= 0.92758999785793
- ❖ precision= 0.9774509803921569
- ❖ F-measure=0.9511188904927508

Results for project3_dataset2.txt: For k=5

- ❖ accuracy=0.6790507364975451(~67.91%)
- ❖ precision=0.5292145354645355
- ❖ recall=0.3708615288220551
- ❖ F-measure=0.428471597122641

- **Choice of parameters:**

  1. Choosing the optimal value of k is best done by inspecting data at first.Cross-validation is another way to retrospectively determine a good K value by using an independent dataset to validate the K value. Historically, the optimal K for most datasets has been between 3-10. After running the code several times for different values of k, k=5 was observed to be an optimal value. Value of k is generally chosen to be an odd number so as avoid any ties.

  2. To reduce overfitting we have used k fold cross validation with value of k equal to 10.

- **Adavantages:**

1. Easy to understand and implement.
2. Training is very fast.
3. It performs well on applications in which samples can have many class labels.

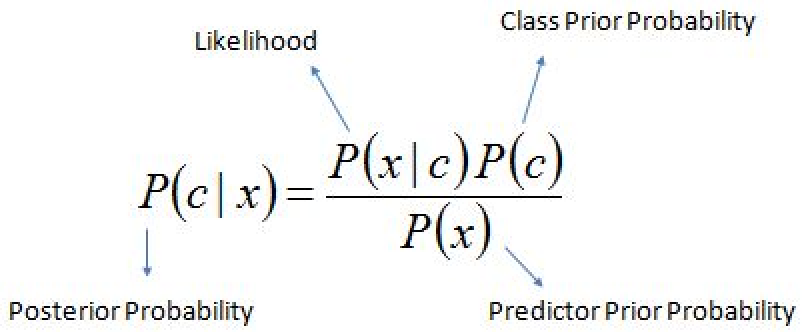
- **Disadvantages:**

1. Lazy learners incur expensive computational costs when the number of potential neighbors which to compare a given unlabeled sample is large.
2. Memory limitation.
3. It is sensitive to the local structure of the data.

Naive Bayes

The Naive Bayesian classifier is based on Bayes' theorem with independence assumptions between predictors. A Naive Bayesian model is easy to build, with no complicated iterative parameter estimation which makes it particularly useful for very large datasets. Despite its simplicity, the Naive Bayesian classifier often does surprisingly well and is widely used because it often outperforms more sophisticated classification methods.

- **Algorithm:** Bayes theorem provides a way of calculating the posterior probability, $P(c|x)$, from $P(c)$, $P(x)$, and $P(x|c)$. Naive Bayes classifier assume that the effect of the value of a predictor (x) on a given class (c) is independent of the values of other predictors. This assumption is called class conditional independence.



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

  - $P(c|x)$ **is the posterior probability of** *class* **(*target*) given** *predictor* **(*attribute*).**
  - $P(c)$ **is the prior probability of** *class***.**
  - $P(x|c)$ **is the likelihood which is the probability of** *predictor* **given** *class***.**
  - $P(x)$ **is the prior probability of** *predictor***.**

- **Implementation:** Following steps are performed for this part:

  Step 1: Dataset file (project3_dataset1.txt or project3_dataset2.txt) and checked for presence of categorical data in the columns. If any column has categorical data then it is encoded by counting unique values in the column and putting those in a set. The index of the set for those values is then used as the encoded values for original categorical data.

  Step 2: The columns containing the categorical data are stored in a list and are treated separately from the continuous data.

Step 3: Using 10 fold cross validation the dataset is split into training dataset and test dataset. After splitting, these are passed to the naive_bayes method.

Step 4: First we have computed the class_prior_probability for both class labels.

Step 5: For Descriptor posterior probabilities we have handled both continuous data and categorical data. In case of continuous data we have computed gaussian probability density functions by using mean and standard deviation values. In case of categorical data we check if the column for which we are computing is in the list of columns of categorical data then we match the value with the test value of the corresponding column for both classes.

Step 6: After computing both posterior and prior probabilities, we have calculated the final probability of the test data row belonging to both the classes and then whichever is higher is assigned to that test data row.

Step 7: After test labels are predicted we have validated it with the original labels of the test data to calculate various metrics: accuracy, precision, recall, f-measure.

Step 8: Final metrics are computed by taking average of all the metrics calculated for all folds.

- **Results:**

Results for project3_dataset1.txt:

- Accuracy= 0.9332393483709274 (~93.32%)
- Precision= 0.9225667534220164
- Recall= 0.8968392718392719
- F-measure=0.9054306031086217

Results for project3_dataset2.txt:

- Accuarcy= 0.7084015275504638 (~70.84%)
- Precision= 0.5848614607786434
- Recall= 0.6154766204727505
- F-measure=0.5926132709817884

- **Advantages:**

1. It requires short computational time for training.
2. It improves the classification performance by removing irrelevant features.
3. It has good performance.

- **Disadvantages:**

1. The naive bayes classifier requires a very large number of records to obtain good results.
2. Less accurate as compared to other classifiers on some datasets.

Decision Tree

Decision tree is a supervised classification algorithm that uses observations about an item (represented in the branches) and finds out about the item's target value (represented in the **leaves**). It is one of the predictive modelling approaches used in statistics, data mining and machine learning.

- **Algorithm:** There are two main parts of the decision tree algorithm. First part is to build a decision tree using the training data. Second part is to use the decision tree for prediction. To build a decision tree, we calculate the gini and information gain of each attribute and split using the attribute which has the maximum information gain.Then this process is recursively done till we get terminal nodes. To predict the class of a test data, we traverse through the decision tree going on the right if the test data has a value greater than the splitting value for that attribute otherwise going to the left if it has a lesser value.We continue this process until we get to the terminal node.The label of terminal node, is the class label for the test data.

- **Implementation:** Following steps are performed for this part:

  Step 1- Dataset file (project3_dataset1.txt or project3_dataset2.txt) and checked for presence of categorical data in the columns. If any column has categorical data then it is encoded by counting unique values in the column and putting those in a set. The index of the set for those values is then used as the encoded values for original categorical data.

  Step 2- 10 fold cross validation is used to split data into training dataset and test dataset by first calculating the size of each fold. Split is done by picking each fold sequentially.

Step 3- Using the training data, build the decision tree. To do this, we first calculate the gini of the entire data using the function. Then for all the attributes,we find the information gain of each and the splitting value of each.We chose the splitting attribute for that node as the attribute which has the highest information gain.We divide the data in two parts.All the data less than the splitting value forms the left data and the rest forms the right data. Then, we recursively do this process on the left and right data till no more splitting can be done.

Step 4- Next step is to predict the class of test data using decision data. For each row of test data,To do this, we check at the root , what is the splitting attribute and splitting value.If the corresponding attribute in test data has a value greater than splitting value, we move to the right child of root else to the left child. Now this node becomes our root and we keep on performing this until we get to terminal node. Then the value of terminal node is assigned as predicted label.

Step 5- We calculate the accuracy, precision, recall and f-measure.

Step 6- Take an average of all the values from all the k-folds for each of the performance measure.

- **Choice of Parameters:**

    1. For preprocessing-We encoded the categorical data into numerical value
    2. To overcome problem of overfitting- we performed 10 fold cross validation in order

- **Advantages :**
    1. Decision Tree requires relatively less data preparation from user
    2. Easy to understand

- **Disadvantages:**
    1. Can have overfitting problem
    2. Can be very large so pruning may be required

- **Results:**

Results for project3_dataset1.txt:

- ❖ Average Accuracy is: 0.9196428571428573 (~91.96%)
- ❖ Average Precision is: 0.910171319908162
- ❖ Average Recall is: 0.8841626891626893
- ❖ Average F-Measure is: 0.8954715307499012

Results for project3_dataset2.txt:

- ❖ Average Accuracy is: 0.608695652173913 (~60.86%)
- ❖ Average Precision is: 0.4388134741075917
- ❖ Average Recall is: 0.4813959578433263
- ❖ Average F-Measure is: 0.4507665180559169

Random Forest

Random 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 of the individual trees. Random decision forests correct for decision trees' habit of overfitting to their training set

- **Algorithm:** Choose *T*—number of trees to grow. Choose $m<M$ (M is the number of total features) —number of features used to calculate the best split at each node (typically 20%)

  For each tree-

  - ❖ Choose a training set by choosing *N* times (*N* is the number of training examples) with replacement from the training set
  - ❖ For each node, randomly choose *m* features and calculate the best split
  - ❖ Use majority voting among all the trees

- **Implementation :** Following steps are performed for this part:

  Step 1- Dataset file (project3_dataset1.txt or project3_dataset2.txt) and checked for presence of categorical data in the columns. If any column has categorical data then it is encoded by counting unique values in the column and putting those in a set. The index of the set for those values is then used as the encoded values for original categorical data.

Step 2- Specify the number of trees that have to be built.

Step 3-10 fold cross validation is used to split data into training dataset and test dataset by first calculating the size of each fold. Split is done by picking each fold sequentially.

Step 4- Create as many decision trees as specified and for each of the decision tree take for training data 63.2% of original training data and the rest as replacements of those 63.2%.This is called sampling with replacement.

Step 5-The final predicted output class for each test data is taken as the mode of predictions by decision trees.

- **Choice of Parameters:**
  1. For preprocessing-We encoded the categorical data into numerical value
  2. To overcome problem of overfitting- we performed 10 fold cross validation in order to avoid overfitting.

- **Results:**

Results for project3_dataset1.txt: Number of Trees=5

❖ Average accuracy is: 0.9471804511278193 (~94.72%)
❖ Average precision is: 0.9442899124600379
❖ Average recall is: 0.9179223912212114
❖ Average f-measure is: 0.9305319797230819

Results for project3_dataset2.txt: Number of Trees=5

❖ Average accuracy is: 0.6231860338243317 (~62.32%)
❖ Average precision is: 0.45880647130647134
❖ Average recall is: 0.38447199730094467
❖ Average f-measure is: 0.40840528054321146

Since Random forest is all about randomly picking up data and features the accuracy , precision, recall and f-measure vary on each run but it maintains its range. Thus the result above is just from one of the many runs of Random Forest Algorithm

Boosting

Boosting is an ensemble technique that attempts to create a strong classifier from a number of weak classifiers.

- **Algorithm :**

Initially, set uniform weights on all the records

At each round:

- Create a bootstrap sample based on the weights (similar to the sample made in random forest)
- Train a classifier on the sample and apply it on the original training set
- Records that are wrongly classified will have their weights increased
- Records that are classified correctly will have their weights decreased
- The error for the classifier (i) is calculated as : $$\varepsilon_i = \frac{\sum_{j=1}^{N} w_j \delta(C_i(x_j) \neq y_j)}{\sum_{j=1}^{N} w_j}$$

  where: w_j= weight of the data point

  ε_i - The sum of all the misclassified weights / Total sum of the weighted matrix
- The classifier's importance is calculated as : $$\alpha_i = \frac{1}{2}\ln\left(\frac{1-\varepsilon_i}{\varepsilon_i}\right)$$
- Then the weight of each record is updated as : $$w_j^{(i+1)} = \frac{w_j^{(i)} \exp\left(-\alpha_i y_j C_i(x_j)\right)}{Z^{(i)}}$$

  That is : if the predicted label and original label is the same then yiCi(xj) is updated as 1

  Else for misclassified label its updated as -1

Final prediction is done by the weighted average of all the classifiers with weight representing the training accuracy. That is to say that the label who has maximum weighted average is the final label for that test sample.

- **Implementation :**Following steps are performed for this part:

  Step 1- Dataset file (project3_dataset1.txt or project3_dataset2.txt) and checked for presence of categorical data in the columns. If any column has categorical data then it is encoded by counting unique values in the column and putting those in a set. The index of the set for those values is then used as the encoded values for original categorical data.

Step 2- Specify the number of trees that have to be built for each k-1 dataset (in this case 9)

Step 3- 10 fold cross validation is used to split data into training dataset and test dataset by first calculating the size of each fold. Split is done by picking each fold sequentially.

Step 4- The data set picked up for train is then assigned weights. The initial weights given to each feature is 1/(number of rows in data).

Step 5- Create as many decision trees as specified and for each of the decision tree take for training data 63.2% of original training data and the rest as replacements of those 63.2%.This is called sampling with replacement. The choice of this 62.3% training data is done in such a way that the features having more weights are selected more randomly to build a decision than the ones which have less weights. More weighted feature signify misclassified results from the previous run.

Step 6- The error of the classifier and its importance(**α**) is calculated after each decision tree is made. Finally we have that number of **α**s as the number of decision trees made for a fold.

Step 7- After each step when a weak classifier is learnt the weights of of the features are increased for misclassified data and decreased for correctly classified data.

Step 8- For predicting the label for a test data the **α** corresponding to each output of the decision tree is taken into consideration and then the majority is picked up as the final label.

- **Choice of Parameters:**
    1. For preprocessing-We encoded the categorical data into numerical value
    2. The initial weights to the features were assigned as 1/(number of rows)
    3. To overcome problem of overfitting- we performed 10 fold cross validation in order to avoid overfitting

- **Results:**

Results for project3_dataset1.txt: Number of Classifiers=5

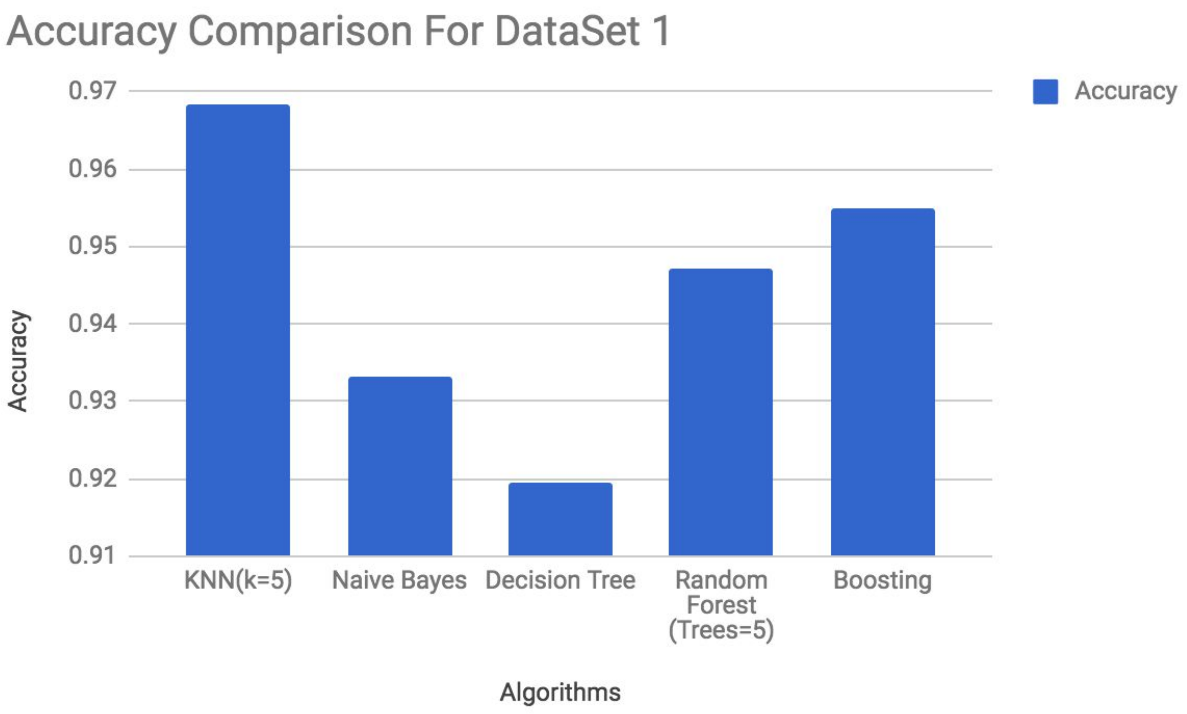
- ❖ Average accuracy is: 0.954832614665381(~95.48%)
- ❖ Average precision is: 0.911239701892063
- ❖ Average recall is: 0.903274916283049
- ❖ Average f-measure is: 0.909930187635418

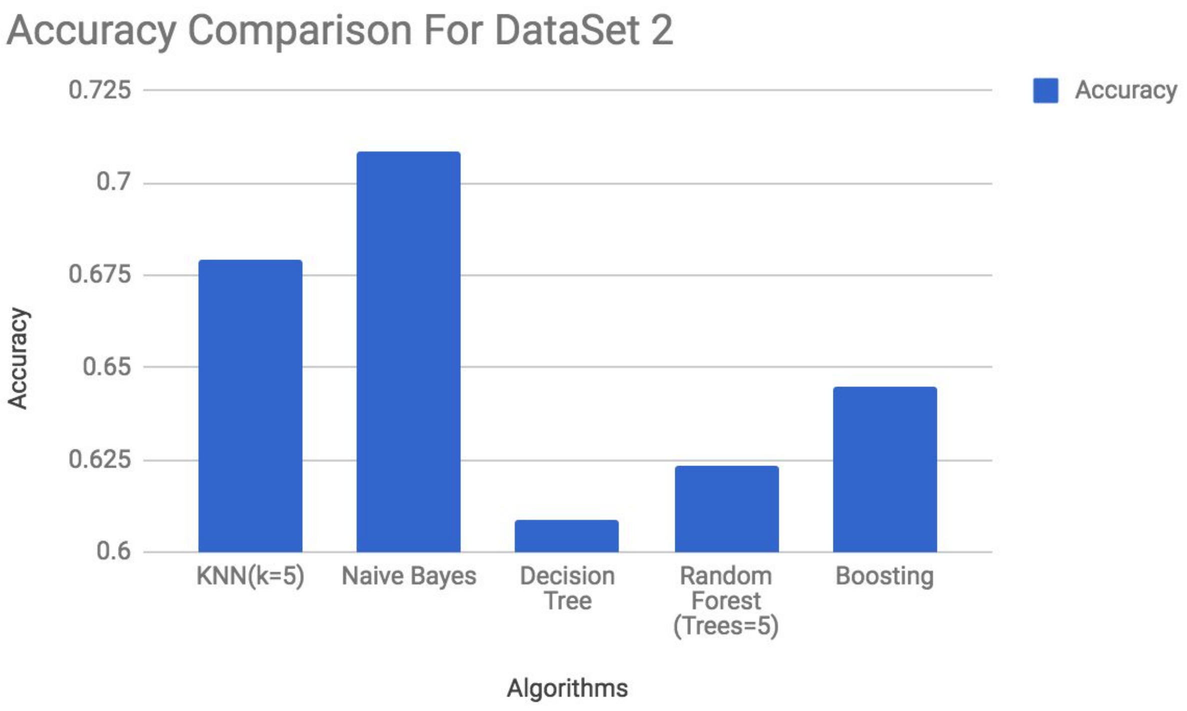
Results for project3_dataset2.txt: Number of Classifiers=5

- ❖ Average accuracy is: 0.6446428571428571(~64.46%)
- ❖ Average precision is: 0.5521739130434783
- ❖ Average recall is: 0.52745180745180745
- ❖ Average f-measure is: 0.53637593415140733

Since Boosting is all about randomly picking up data and features the accuracy , precision, recall and f-measure vary on each run but it maintains its range. Thus the result above is just from one of the many runs of Boosting Algorithm.

Comparison


Accuracy Comparison For DataSet 1
Accuracy
0.97
0.96
0.95
0.94
0.93
0.92
0.91
KNN(k=5)
Naive Bayes
Decision Tree
Random Forest (Trees=5)
Boosting
Algorithms
Accuracy


Accuracy Comparison For DataSet 2
Accuracy
0.725
0.7
0.675
0.65
0.625
0.6
KNN(k=5)
Naive Bayes
Decision Tree
Random Forest (Trees=5)
Boosting
Algorithms
Accuracy

- **Comparison between k-NN, Naive Bayes and Decision Tree**

Parameter	**k-NN**	**Naïve Bayes**	**Decision Tree**
Deterministic/Non-Deterministic	Non- deterministic	Non-deterministic	Deterministic
Effectiveness	Small Data	Huge data	Large data
Speed	Slow for larger data	Faster than k-NN	Faster
Dataset	It can't deal with noisy data	It can deal with noisy data	It can deal with noisy data
Accuracy	Provides high accuracy	For obtaining good results it requires a very large number of records	High accuracy

- **Comparison between bagging and boosting**

1. Both are ensemble methods to get N learners from 1 learner but while they are built independently for Bagging, Boosting tries to add new models that do well where previous models fail.
2. Both generate several training data sets by random sampling but only Boosting determines weights for the data to tip the scales in favor of the most difficult cases.
3. Both make the final decision by averaging the N learners (or taking the majority of them) but it is an equally weighted average for Bagging and a weighted average for Boosting, more weight to those with better performance on training data.