CSE 601: Data Mining and Bioinformatics

Project 3: Classification Algorithms

Submitted by

Bhagutharivalan Natarajan Muthukkannu (50314871)

Harish Kannan Venkataramanan (50316999)

Praveen Mohan (50321225)

K-fold Cross Validation

Cross validation is a statistical method used to estimate the performance (or accuracy) of machine learning models. K fold cross validation splits the whole dataset into k partitions. At each iteration one partition is used as a test set and the remaining k-1 partitions are taken as a train set. The performance metric generated at each iteration is averaged out at the end of iteration. K- fold cross validation is very useful when handling imbalanced datasets. It helps to avoid underfitting of the model.

K -Nearest Neighbors

KNN is a lazy learner classification algorithm.

Pseudo Code

1. The data is split into train data (where the model is trained) and test data (where the used for testing).
2. The features (columns) are normalized if required or strings converted into numerical for easy processing if required.
3. K is obtained from user input where K is the number of nearest points to be considered.
4. The Euclidean distance is calculated between all the points in the test and the train dataset.
5. The K number of nearest data points are taken for categorizing the test data point.
6. The voting weights (1/d2) are calculated by the distance between the train and test data points.
7. The test data point is labelled or categorized by the majority of the train data labels near them.

8. The efficiency of the classification is calculated by f1 score, accuracy, precision and recall.

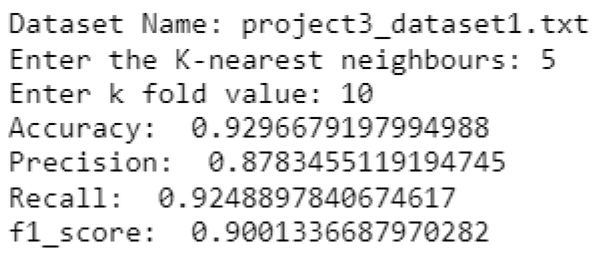
Advantages

1. KNN algorithms is a lazy learner's algorithm and does not involves expensive computation for training the models.
2. Simple and efficient to implements.

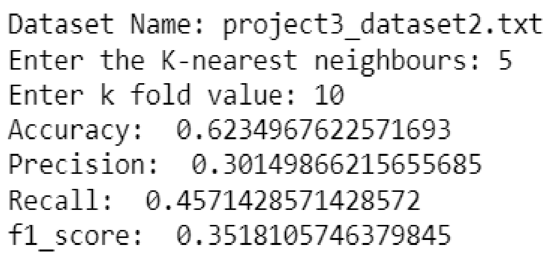
Disadvantages

1. There is not standard method of defining K value.
2. Needs feature scaling.
3. Doesn't work very well for large datasets and high dimensional data sets.

Results



```
Dataset Name: project3_dataset1.txt
Enter the K-nearest neighbours: 5
Enter k fold value: 10
Accuracy:  0.9296679197994988
Precision:  0.8783455119194745
Recall:  0.9248897840674617
f1_score:  0.9001336687970282
```



```
Dataset Name: project3_dataset2.txt
Enter the K-nearest neighbours: 5
Enter k fold value: 10
Accuracy:  0.6234967622571693
Precision:  0.30149866215655685
Recall:  0.4571428571428572
f1_score:  0.3518105746379845
```

Naïve Bayes

Naïve Bayes is Bayesian Classifier algorithm which predicts classes based on membership probabilities using Bayes Theorem. It estimates Class Posterior Probability for each samples of the test data and assigns the class which has the highest class posterior probability. This classifier assumes that each attribute is independent while calculating probability values. To find the class posterior probability we need to compute the following probability values.

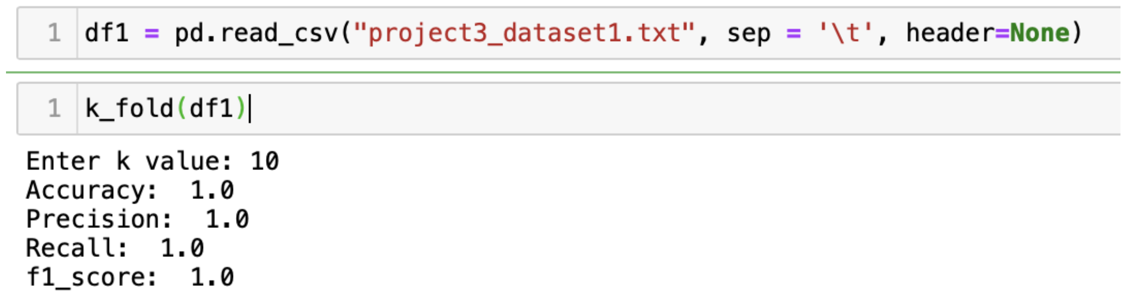
1. Class Prior Probability { $P(H_i)$ }
2. Descriptor Posterior Probability { $P(X \mid H_i)$ }
3. Descriptor Prior Probability { $P(X)$ }

$$\mathbf{P(H_i \mid X) = P(H_i) * P(X \mid H_i) / P(X)}$$

Algorithm

1. Calculate Class Prior Probability for the entire training dataset.
2. Calculate Descriptor Posterior Probability for each column with respect to the class variable.
3. For each sample in the test data calculate Class Posterior Probability for each class and assign the sample to the class which has the highest probability.

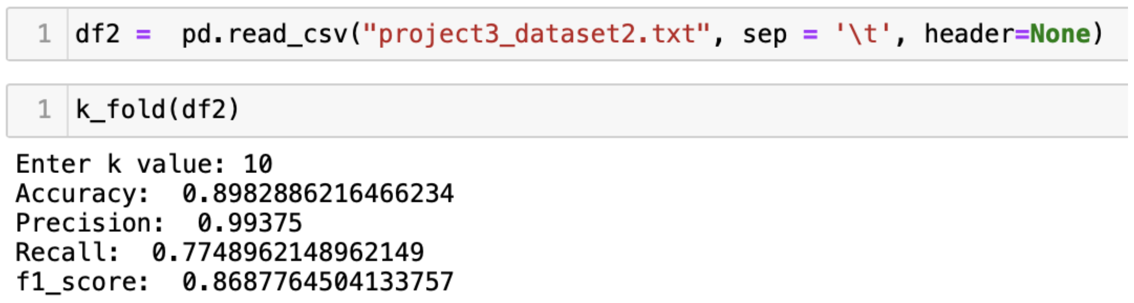
Results



```
df1 = pd.read_csv("project3_dataset1.txt", sep = '\t', header=None)
```

```
k_fold(df1)
```

```
Enter k value: 10
Accuracy:  1.0
Precision:  1.0
Recall:  1.0
f1_score:  1.0
```



```
df2 =  pd.read_csv("project3_dataset2.txt", sep = '\t', header=None)
```

```
k_fold(df2)
```

```
Enter k value: 10
Accuracy:  0.8982886216466234
Precision:  0.99375
Recall:  0.7748962148962149
f1_score:  0.8687764504133757
```

Decision Tree

A decision tree is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility.

Pseudo Code

1. The number of folds to be split is taken as an input from the user(k-value).
2. Now the dataset is split into train and test data by the k-fold function.
3. The possible split values for each column are found by taking the unique values in each column.
4. The gini impurity is calculated for each possible split value and the split value with the lowest gini impurity is taken as the best split condition.
5. Now the whole dataset is divided into 2 partitions (child nodes) based on the split condition.
6. The empty dictionary is created and key of the dictionary has a format – column_number <= split_value(example: 2 <= normal).
7. Now the child node becomes parent node, when it is impure and size of child node >= minimum number of samples and the depth of the tree<= maximum depth to be traversed.
8. If the child node violates any of the condition mentioned above then the child node is considered to be leaf node and the leaf node will have the class label.
9. Class label of the leaf node is added to the dictionary.
10. The process is recursively done till we get the leaf node for each branch.
11. K metrics from the k fold cross validation is averaged out to get the final metrics.

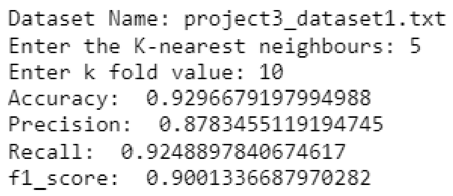
Pros

1. They are inexpensive to construct and extremely fast at classifying unknown records.
2. The decision tree induction is a non-parametric approach for building classification models meaning not requiring any assumptions regarding the type of probability distributions satisfied by the class and other attributes.
3. They perform well on learning discrete-valued functions.
4. They are quite robust to the presence of noise, especially when with methods that avoids overfitting.
5. They do not get affected by presence of redundant attributes too.

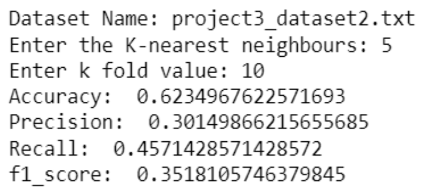
Cons

1. The Decision tree models generally overfit the data and poorly predicts the test data.
2. Decision tree performs badly on the continuous data and works well for discrete data.
3. This algorithm makes decision trees susceptible to high variance if they are not pruned.
4. Due to top-down approach, the number of records at leaf node may be too small to make a statistically significant decision about the class representation of the node, also known as data fragmentation. The solution is setting a maximum depth limit.
5. Subtree replication can cause the decision tree to become more complex than necessary also adding to difficulty in interpretation.

Results



```
Dataset Name: project3_dataset1.txt
Enter the K-nearest neighbours: 5
Enter k fold value: 10
Accuracy:  0.9296679197994988
Precision:  0.8783455119194745
Recall:  0.9248897840674617
f1_score:  0.9001336687970282
```



```
Dataset Name: project3_dataset2.txt
Enter the K-nearest neighbours: 5
Enter k fold value: 10
Accuracy:  0.6234967622571693
Precision:  0.30149866215655685
Recall:  0.4571428571428572
f1_score:  0.3518105746379845
```

Random Forest

Random forest is an ensemble learning technique where weak learners are converted into strong learners.

Pseudo Code

1. Number of trees(n) to be grown and number of features at each split(m) is taken as a input from the user.
2. n number of decision trees are generated where m features are randomly considered at each split.
3. Each sample from the test data is fed to the n trees and the output from n trees is averaged out.

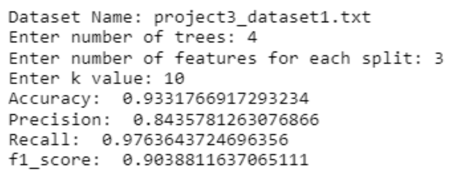
Pros

1. Random forest is the state-of-the-art technique and it produces accurate classifier.
2. Thousands of input features can be handled to predict correct labels without deletion of any column
3. It reduces the overfitting and variance of the model without increasing the bias.
4. Random forest doesn’t necessarily need to be cross validated.

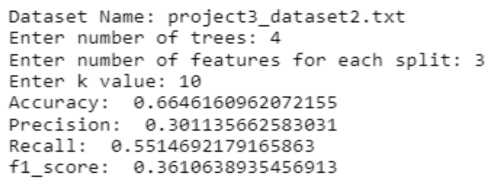
Cons

1. Nonlinear relationship between dependent and independent variable cannot be captured with the random forest.
2. Large number of trees can make the algorithm slow.

Results



```
Dataset Name: project3_dataset1.txt
Enter number of trees: 4
Enter number of features for each split: 3
Enter k value: 10
Accuracy:  0.9331766917293234
Precision:  0.8435781263076866
Recall:  0.9763643724696356
f1_score:  0.9038811637065111
```



```
Dataset Name: project3_dataset2.txt
Enter number of trees: 4
Enter number of features for each split: 3
Enter k value: 10
Accuracy:  0.6646160962072155
Precision:  0.301135662583031
Recall:  0.5514692179165863
f1_score:  0.3610638935456913
```