CSE 601 – Data Mining and Bioinformatics

Project 3: Classification Algorithms

Team members:

Kiran Ketan Nevrekar, 50336915, kiranket

Revathy Narayanan, 50336857,revathyn

Saranya Saravanan, 50314931, saravan2

Objective:
The aim of this project is to implement four classification algorithms using the given two dataset. And also to evaluate the performance of all the four algorithms using 10 cross validation in terms of accuracy, precision, recall and F1 score.

Introduction:

Classification Algorithms: Classification is supervised machine learning algorithms which learns from input data, builds a model which is then used on test data to check for accuracy. There are various types of classification algorithms and they are logistic regression, K Nearest Neighbors, Naive Bayes, Decision trees etc. And they are used in areas like sentiment analysis, credit card/loan application, Email Spam classification, Image/Document classification, etc.

Performance analysis: The performance of classification algorithms are done using 10 cross validation in terms of accuracy, precision, recall and F1 score. K Cross Validation is a technique for evaluating machine learning models by training them on subsets of training data and testing them on complementary subsets of data. This helps in analyzing how the built model generalises to an independent dataset. A confusion matrix is a table that is often used to describe the performance of a classification model (or "classifier") on a set of test data for which the true values are known. And using the confusion matrix we can find the following.

Accuracy :
True Positive = TP
True Negative = TN
FalsePositive = FP
FalseNegative = FN

Accuracy = (TP+TN) / (TP+TN+FP+FN)
Precision = (TP) / (TP + FP)
Recall = (TP) / (TP + FN)
F-1 measure = 2 x [(PXR)/(P+R)]

Algorithms Implementation:

Nearest Neighbor Algorithm : K Nearest Neighbor is a classification algorithm that classifies data based on the points that are similar to it and can be solved by both classification and regression problems.

Implementation :

- Read the file to input the data frame
- Splitting the data frame to store the features and labels in separate dataframes “features” and “train_labels” respectively.
- Check if the data contains any attribute with type as string, object and for transforming the categorical variable we perform one hot encoding by finding the number of unique elements in the particular attribute and forming a new column by setting its corresponding value as 1 and rest others as 0. Append these newly framed columns to the features dataframe and drop the original attribute.
- Normalizing the pre processed data by using min max normalization.
- Set the k value. I.e the number of neighbors.
- For performing the 10 fold cross validation, we split the features dataframe and the labels dataframe into 10 folds of equal size.
- By iterating through the folds, for each iteration we will hold one particular fold as test data and the remaining 9 folds (incase of 10 fold CV) as train data and perform K-Nearest Neighbor classification for each iteration. We will also be storing the accuracy, precision, recall and F1Score of each model to compute the average performance metrics later at end.
- For KNN implementation:
    - The test and train data are passed as arguments.
    - The Euclidean distance is computed between each row in test data with each row in train data and stored in a matrix.
    - Now for each row in test data sort its distance to all the train data
    - Consider the first K distances and retrieve the class labels for the corresponding train data row.
    - Check for the maximum number of labels that have been classified into the same class.
    - The same class label as of the maximum voting is assigned to the test data row.
    - From the returned predictions and the Available test labels the performance metrics are computed.
- The performance metrics formulations are stated above.
- The overall accuracy, precision, recall and F1score are computed as the average of the values returned by each fold of the cross validation

Packages used : Numpy, pandas, matplotlib

Result:
For the dataset1 the KNN classification model predicts labels with an accuracy of 96% and for dataset2 it gives accuracy of 68%. This also shows that the model may not necessarily perform well in all the dataset. The performance of the model depends on the dataset and features as KNN is sensitive to outliers and noise

Advantages:

- Easy to implement.
- KNN runs faster than any other classification algorithm.
- It is a lazy learner algorithm.

Disadvantages:

Sensitive to noisy data, missing values and outliers
Does not work well with large dataset:
Requires normalization to be performed

Output Values:

For Project dataset 1:
K=3

```
average accuracy: 96.48496240601503
average pecision: 96.65922005322156
average recall: 95.6692826425728
average f1score: 0.9615204204223708
```

For Project dataset 2:
K=3

```
average accuracy: 68.17761332099907
average pecision: 64.10640604067372
average recall: 63.36401201246589
average f1score: 0.6371042740391712
```

Naïve Bayes Algorithm : Naive Bayes is a probabilistic classification algorithm used to classify the given data by assuming the presence of a particular feature in a class that is unrelated to the presence of any other feature. This classification model is easy to implement and is widely used. This classifier uses bayes theorem and its represented as follows:

Bayes theorem = > P(A/B) = (P(B/A) P(A)) / P(B)

Using Bayes theorem, the probability of A happening is found, given that B has already occurred. Here, B is the evidence and A is the hypothesis. The assumption is that the

presence of one feature does not affect the other. And this is the essence of Naive Bayes classifier.

Implementation :

Naive Bayes classifier is implemented as follows,

1. The data file is obtained from the user through the console. The last column of the data file is the groundtruth labels which is stored in a variable so that it can be used to compare the results with groundtruth labels.
2. Firstly, we convert categorical/nominal data to numerical data. We have created a dictionary to store the categorical data with their corresponding numerical values.
3. A method to split the dataset accordingly for cross validation is introduced. The dataset is split into 10 groups and at every iteration one group is considered as test and the rest of the nine dataset groups are considered as train for this model.
4. Next, we have calculated the total probability for 1 and 0 class labels. And also introduced two separate functions to handle continuous and categorical/nominal data.
5. Unlike nominal data, we cannot count different value counts for continuous data and for that we use Gaussian Distribution model. A Gaussian distribution is usually chosen to represent the class-conditional probability for continuous attributes and this is characterized by mean and variance.
   The formula is as follows:

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

   We perform the above given method, separately, for zero and ones of continuous values in the given data. These results are stored in two separate lists for zero and one class. This is then used later in another function to combine the results of nominal/categorical results and continuous results. We have used the Laplacian method to handle the case where the class probability might be zero or one.
6. In another method we have combined the results of nominal and categorical by multiplying the zero probability values of continuous and categorical/nominal for the test tuples. Performing the same for one probability. Now, we have two probability values, probability value of one and probability value for zero. These

values are compared and whichever value is higher the test tuple is classified to that class label.

7. This is then compared with the ground truth class labels. And performance metrics are calculated in a separate function. The accuracy, precision, recall and f1 score values are discussed in the following sections.

Packages used : Numpy, pandas

Result / Advantages and Disadvantages: The accuracy for the dataset 1 is approximately 93 and for dataset 2 is approximately 70. And from this we can see that the Naive Bayes classification model that we have built works well for dataset 1 than dataset 2. Here dataset 1 has only numerical values and dataset 2 has one categorical/nominal column.The advantages of this classifier based on our findings are that : 1. This classifier is easy to implement. 2. Performs well in multi-class prediction. 3. When the assumption for independence holds, it performs fast compared to other algorithms and also delivers higher accuracy. And the disadvantages would be that the assumption that all features would be independent may not hold true in real life, makes it complicated and delivers less accuracy in this case.

Accuracy, Precision, Recall, and F-1 measure :

For Project dataset 1:

```
Average Accuracy is: 93.31453634085214
Average Precision is: 91.31925736801898
Average Recall is: 90.44426356826324
Average Fscore is: 0.9078725938618135
```

For Project dataset 2:

```
Average Accuracy is: 70.34690101757633
Average Precision is: 57.09275640080593
Average Recall is: 61.59396451501714
Average Fscore is: 0.5867396458138147
```

Decision Tree Algorithm : Decision Tree is a classification algorithm that is used to create a training model which can be used to predict class labels of the given test data by simply learning decision rules. This algorithm can also be solved by regression and classification problems. We build a decision tree with the given data that has more than one attribute present in it and hence it is necessary to evaluate the importance of each

of them thus placing the most important one at the root and form the tree by splitting the nodes further. The best split is calculated by something called the Gini Index and with this as we move down the tree, the impurity and uncertainty decreases. This measures the probability of a variable being wrongly classified when it is randomly chosen.

Implementation :
Decision Tree Algorithm is implemented as follows:

1. The data file is obtained from the user through the console. The last column of the data file is the groundtruth labels which is stored in a variable so that it can be used to compare the results with groundtruth labels.
2. Firstly, we convert categorical/nominal data to numerical data. We have created a dictionary to store the categorical data with their corresponding numerical values.
3. A method to split the dataset accordingly for cross validation is introduced. The dataset is split into 10 groups and at every iteration one group is considered as test and the rest of the nine dataset groups are considered as train for this model.
4. Now, to start building decision tree we need to find out the root node and we do this by calculating the gini index for all the features in the given data. We set threshold values for features and if they are lesser or equal they go to the left else right node.
   The formula for Gini Index is as follows:

$$Gini = 1 - \sum_{i=1}^{C} (p_i)^2$$

5. We recursively build a decision tree by calculating gini index and we perform split at nodes by considering the lowest gini index value of all the features' gini index values. And we do this recursively for the nodes till they split and the nodes are identified as leaf nodes i.e when the value of gini index is 0, it is recognized as a leaf node.
6. We test the test data by traversing through the tree till it reaches a leaf node and return the class label of that node.

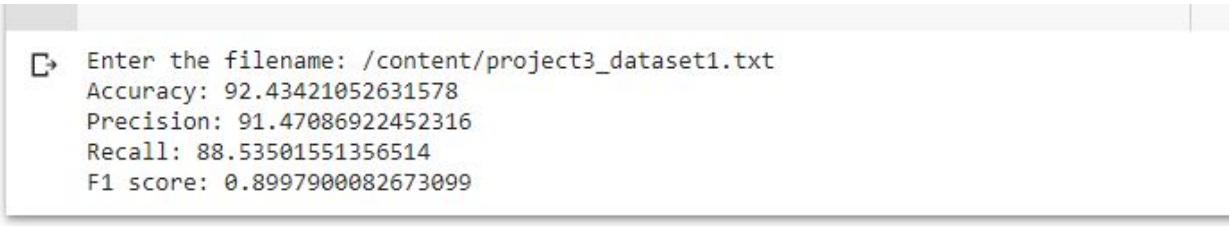
Packages used : Numpy, pandas, matplotlib

Result / Advantages and Disadvantages: The accuracy for the dataset 1 is approximately 92 and for dataset 2 is approximately 64. And from this we can see that

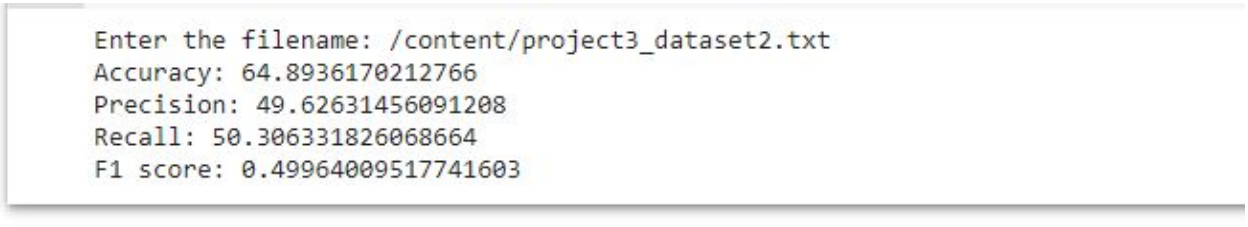
the Decision Tree algorithm that we have built works well for dataset 1 than dataset 2. The advantages of this algorithm based on our findings are that: 1. They are easy to understand and implement. 2. Can handle both categorical and numerical data. 3. Easy to visualize decision trees. And the disadvantages are as follows. Overfitting might be an issue sometimes and has to be pruned because of that. Follows greedy approach and selects local optimum solutions at each node.They are not robust to noise.

Accuracy, Precision, Recall, and F-1 measure:

For Project dataset 1:



For Project dataset 2:



Random Forests Algorithm : Decision Tree forms a building block for random forest and is an intuitive model. It consists of a large number of individual decision trees which gives out class predictions and the class with the most number of votes becomes the model prediction. It operates as a group and outperforms any individual models. We have built a Random Forest algorithm based on our decision tree algorithm that we have implemented before.

Implementation:
Random Forest algorithm is implemented as follows:

1. The data file is obtained from the user through the console.
2. The number of trees and number of features are taken as input from the user through the console.
3. The last column of the data file is the groundtruth labels which is stored in a variable so that it can be used to compare the results with groundtruth labels.

4. We convert categorical/nominal data to numerical data. We have created a dictionary to store the categorical data with their corresponding numerical values.
5. A method to split the dataset accordingly for cross validation is introduced. The dataset is split into 10 groups and at every iteration one group is considered as test data and the rest of the nine dataset groups are considered as train data for this model.
6. We create random subset of the training dataset by using np.random.choice. We then find a root node to split the dataset in order to build decision tree.
7. Based on the input given for the number of features, the function find_node_split selects a subset of features while splitting the node. We repeat the same for the number of trees given by the user.
8. We store results from all the trees in finalPredictedValues and the class label which gets the majority vote is considered to be the final prediction of the algorithm.

Packages used : Numpy, pandas

Result / Advantages and Disadvantages: The accuracy for the dataset 1 is approximately 94 and for dataset 2 is approximately 62. And from this we can see that the Random Forest algorithm that we have built works well for dataset 1 than dataset 2. The advantages of this algorithm based on our findings are that: 1. Works well on non-linear data. 2. It's powerful and gives higher accuracy. 3. Can be used as classification as well as regression. The disadvantages of Random forest are as follows: 1. It's slightly complex 2. Have to manually provide the number of trees and is not in control of everything that it can do. 3. Does a better job in classification than regression.

Accuracy, Precision, Recall, and F-1 measure :

For Project dataset 1:

```
Enter the filename: /content/project3_dataset1 (2).txt
Enter the number of trees: 3
Accuracy: 94.02255639097744
Precision: 94.13948500136655
Recall: 89.5506763233436
F1 Measure: 0.9178776358850734
```

For Project dataset 2:

```
Enter the filename: /content/project3_dataset2 (2).txt
Enter the number of trees: 3
Accuracy: 62.98797409805735
Precision: 48.64407011117538
Recall: 40.07417290312027
F1 Measure: 0.43945209240282956
```

Kaggle Competition:

- Initially the data was split to hold out a validation set using the 80:20 split.
- The whole model was initially built using the 80% percent data i.e(320 data points).

Step1:

The data was initially fit with the following classification models:

- nearest neighbor
- decision tree
- Naïve Bayes
- SVM
- Bagging
- AdaBoost
- random forests
- Gradient Boosting

Each of the model's F1score ranged between (0.6 to 0.81)

To increase the performance of the model the following techniques were performed:

- Performed gridsearchCV for each of the models to find the best parameters.

- Performed bootstrapping.
- Performed Principal Component Analysis for feature reduction and to capture the variance.

Step 2:

- GridSearchCV was performed using 5 folds.
- A list of values for the parameters were specified to fit the model with the Xtrain and Ytrain data.
- From the output of the GridSearchCV the best parameters were retrieved using the function best_params_
- PCA was performed on the train data. The ideal number of components was retrieved using the explained_variance_ration_ function.
- A pipeline was created to transform the data by passing the retrieved best number of components from PCA and the model was generated after tuning the hyperparameters.
- These best parameters from the gridsearchCV were used to fine tune the model further.
- The pipeline was fit to execute the functions in the pipeline with the train data.
- The subset of the 80 percent train data was used as test data here.
- The prediction was made using this test data and corresponding F1Score was calculated.

To validate the prediction, the step 2 was iterated for 3 loops. Each iteration splits and takes a set of data as train and test and performs the entire step2 to get an individual F1Score.

The individual F1Scores were stored in a list and the average F1Score was computed.

This Average F1Score was used to choose the best fitting models.

Average F1Scores of models:

```
Adaboost: 0.852541138779671

KNN: 0.8643464119654596

SVM: 0.8310169508120137

Gradient Boosting method: 0.31661998132586366

Random Forest Classifier: 0.4726885776066104
```

To validate the results, we are performing predictions on the test set that we held initially.

It gives the following F1Scores:

```
fscore Adaboost 0.7874015748031497

fscore KNN 0.845360824742268

fscore SVM 0.2711864406779661

fscore Random Forest 0.7142857142857144

fscore Gradient Boosting 0.5205479452054794
```

Only for certain samples of the data the SVM model was performing well. However KNN, Adaboost and Random forest have better performances. Gradient boosting method is not performing well with this data set.

Since each data point is important, Finally the entire data is used to fit the model so as not to lose any data and the predictions are made with the test data, written to a csv file in specified format and submitted to Kaggle.