Data Mining and BioInformatics
Project 3: Classification Algorithm

Abhav Luthra(50288904) - Krishna Sehgal(50291124)
Tanmay Pradeep Singh(50291086)

Classification

It is the process of finding a model that describes a class attribute as a function of the values of other attributes. The goal is to identify category of a new observation from a set of categories on the basis of a training set of data containing observations whose categories membership is known.
The objective of classification task is to evaluate the prediction ability of a model. A training data set is divided into training set and test set. The model is trained on the training set and prediction is done on the test set. The predicted labels for the test set are compared with their true labels to evaluate the accuracy of the model.

k - Fold Cross Validation

We are performing a 10-fold cross validation in order to get a good estimate of the accuracy measures of the algorithm.

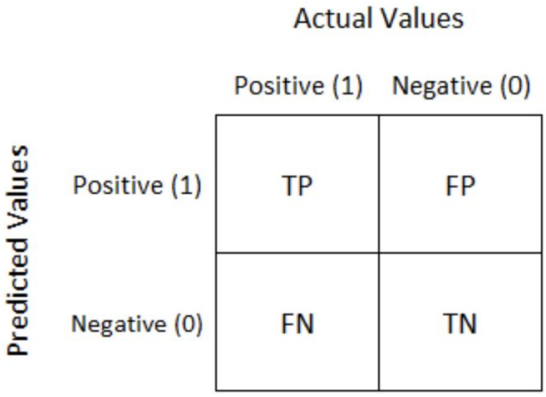
Below is the way this is implemented:

- We first shuffle the dataset and then identify the size of the test data set required for a 10-fold validation.
- We then extract this test set and move the remaining elements into another matrix and call them as train set.
- We perform regular algorithm on this test and train dataset pairs and compute the performance metrics.
- We next go on to the next fold, i.e, the next set of test data set elements are extracted and now the remaining data is the train data set, and algo is performed on this.
- The process is repeated 10 times until different segments of the dataset is considered as test dataset and performance metrics are evaluated for each of the folds.
- Finally, we calculate the average performance metric across the 10-folds.

Metrics for Performance Evaluation

Confusion Matrix

The Confusion matrix is used for finding the correctness and accuracy of the model. It is used for Classification problem where the output can be of two or more types of classes.



Terms associated with Confusion matrix

1. True Positives (TP): True positives are the cases when the actual class of the data point was 1(True) and the predicted is also 1 (True).
2. True Negatives (TN): True negatives are the cases when the actual class of the data point was 0(False) and the predicted is also 0 (False).
3. False Positives (FP): False positives are the cases when the actual class of the data point was 0 (False) and the predicted is 1 (True). False is because the model has predicted incorrectly and positive because the class predicted was a positive one (1).
4. False Negatives (FN): False negatives are the cases when the actual class of the data point was 1 (True) and the predicted is 0 (False). False is because the model has predicted incorrectly and negative because the class predicted was a negative one (0).

Widely used Metric

Accuracy = TP + TN / TP + TN + FP + FN
Precision = TP / TP + FP
Recall = TP / TP + FN
F-score = 2 * Recall * Precision / Recall + Precision

K Nearest Neighbor

It is a Supervised Machine Learning Algorithms as the target variables are known. In KNN we use the training records to predict class labels of unseen data i.e test data. In order to perform KNN algorithm we require a value of K to determine K nearest records known records that are close to the record whose class label is need to be identified. Also a set of records having a known class label is required. A distance

metric is also required to compute the distance between the records (known and unknown class label record). Following are the steps to perform KNN algorithm.

Flow of K Nearest Neighbor:

1. Perform k- fold (taken as 10) cross validation on the dataset to divide data into training and testing data where training is done on k-1 partitions and testing is done on the remaining one partition iteratively i.e in the first iteration, first partition is taken as test data and the rest is taken as training data. In the second iteration, the second partition is taken as testing data and the first and rest of partitions are taken as training data and this way it is done for all k partitions.
2. Determine the parameter K which represent the number of nearest neighbors.
3. Calculate the distance between the query instance (unknown class label record) and all the training samples whose class label is known.
4. Sort the distance and determine nearest neighbor based on the K-th minimum distance.
5. Gather category (class label) of the nearest neighbors.
6. Perform majority/ voting of the category of nearest neighbor as the prediction value of the query instance.
7. Perform voting for the unknown class label based on the top K nearest class labels to determine the category of unknown label.
8. Compare the predicted value of label to the actual class label and find the Accuracy, Precision, F-score and Recall and perform this step on each iteration of k - cross validation(k =10).
9. Find average of Accuracy, Precision, F-score and Recall for each of the k iterations performed on data set while performing k-cross validation.

NOTE: The value 'K' is different from 'k'. 'K' represents the nearest neighbors from the query instance whereas k represent parameter for cross validation that represent number of partitions performed on the dataset.

Handling continuous/categorical attributes

Since Euclidean distance is used as a distance metric, numerical attributes do not need any manipulation in terms of how they are represented. However, for categorical attributes, a simple method to calculate their distance is to assign 0 when the values for that attribute exactly match or 1 when they don't. This can be done using Label Encoder Method. Other, more granular distance schemes may also be used which would assign different weights amongst different values. For e.g. dist(green, red) = 0.6 while dist(green, blue) = 0.8 and so on. In our implementation, we have used former (0/1 method)

Also, while performing KNN, there is a need to take care of issues that can result in a poor training of model. They are:

1. The value of K should be taken wisely, If it is taken too small it can add noise points. If taken too large may include points from other classes.

2. The attributes should be scaled in order to prevent domination of one attribute while measuring distance.

Parameters

The value of K determines the granularity of the boundary between predicted classes. This usually is data dependent, but generally, a higher value of K makes the algorithm robust against outliers. This is because we take more votes into account, but also blurs the boundaries between classes. A lower value of K will give crisper class boundaries but will also suffer from local minima.

Result

We ran the model for K = 2 to 15. Best results for dataset1 was observed on K = 5 for dataset 1 and K = 11 for dataset 2

The average metrics for accuracy, precision, recall and f1-measure for 10 folds on both datasets are given below as follows:

	Project3_dataset1.txt	**Project3_dataset2.txt**
Average Accuracy	**0.919172932330827**	**0.5773049645390071**
Average Precision	**0.924042397660818**	**0.34244311244311243**
Average Recall	**0.850248753707780**	**0.2438010892616156**
Average F Score	**0.882820059070059**	**0.27903458824739363**

Pros of KNN

1. KNN is a simple algorithm and hence it is easy to interpret the prediction of unknown class labels.
2. It is non parametric, so it does not make assumptions about the underlying data pattern.
3. It can be used for both Classification and Regression problems.
4. The training step is much faster as compared to other Machine Learning algorithms.

Cons of KNN

1. They are are lazy learners as KNN does not have a training step. All data points will be used only at the time of prediction. With no training step, prediction step is costly. An eager learner algorithm eagerly learns during the training step.

2. KNN is computationally expensive as it searches the nearest neighbors for the new point at the prediction stage.
3. It requires high memory requirement as KNN has to store all the data points.
4. It is sensitive to outliers due to which the accuracy is impacted by noise or irrelevant data.

Comparison

1. Naive bayes classifier is parametric whereas KNN is non-parametric.
2. If training data is much larger than the number of features , KNN performs better as compared to SVM. SVM outperforms KNN when there are a large number of features and lesser training data.
3. Decision tree is faster as compared to KNN because of its expensive real time execution.

Naive Bayes

Naive Bayes is a statistical classifier which is based on Bayes Theorem. This classifier predicts the class membership probabilities such as the probability that a tuple belongs to a particular class. This algorithm assumes that all the attributes of a tuple are independent of each other and have no correlation whatsoever. This assumption is called class-conditional independence.

Let *H* be the hypothesis that a tuple *X* belongs to a class label *C*. For classification, we look for the probability that the tuple *X* belongs to class *C* given that we know the attributes of that tuple. This is also called as the *posterior probability* and is given by *P(H|X)*.

Terms used in Naive Bayes

- *P(H|X)* : Probability of hypothesis *H* holding true that a tuple *X* belongs to class *C* given that we know the attributes of *X*. (Posterior probability) (a.k.a posteriori probability of *H* conditioned on *X*).
- *P(H) :* It is the prior probability of hypothesis H holding true which is independent of the data tuples.
- *P(X|H)* : It is the probability that there exists a tuple *X* such that it satisfies the hypothesis *H* of belonging to class *C*. (Posterior probability of *X* conditioned on *H*)
- *P(X) :* It is prior probability of existence of tuple *X*. We have used this formula to perform classification: $P(H|X) = (P(X|H) * P(H)) / P(X)$

Flow of Naive Bayes

1. In the above formula *P(X)* is constant for all the classes and hence we have ignored this term in our implementation.
2. First, we divide dataset into training data and test data.
3. For a tuple in the training data with class label *Ci*, if we come across a numerical column (attribute), we calculate mean and variance for that column of the dataset. In case of categorical column, we do not perform this calculations for that tuple.

4. The posterior probability for each tuple *X* i.e *P(X|Ci)* or *P(X|H)* in the test data is calculated depending on the attribute type. For all the numerical attributes we assume normal distribution and calculate probability density given by the formula:

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where, **μ** = Mean, ***σ*** = Standard Deviation 2, ***σ*** = Variance

We use the mean and standard deviation values that we already calculated for each column, to calculate this probability. If we come across a categorical column, we calculate the number of times the value in that column has appeared in the dataset for the *Ci* and divide by the count of total number of *Ci*s in the dataset. We ignore the last column which represents the class label for our training data .

● The posterior probability *P(X|Ci)* is the multiplication of the calculated probabilities over all the attributes to the prior probability *P(Ci)* or *P(H)*.

Result

The average metrics for accuracy, precision, recall and f1-measure for 10 folds on both datasets are given below as follows:

	Project3_dataset1.txt	**Project3_dataset2.txt**
Average Accuracy	**0.8172619047619047**	**0.6922531369339879**
Average Precision	**0.6921863639451298**	**0.5698839065131635**
Average Recall	**0.9083815936161475**	**0.5293508280505185**
Average F Score	**0.7823839221872128**	**0.5391707296417596**

Pros of Naive Bayes

1. It is easy to understand and implement Naive Bayes Classifier.
2. If the conditional independence assumption actually holds true, then this classifier converges faster than other classifiers, hence will require less training data.
3. It Handles continuous as well as discrete data.

4. It makes probabilistic predictions.
5. It is efficient when dealing with large databases.

Cons of Naive Bayes

1. This classifier doesn't perform well if the dataset have correlated attributes. Hence it is called 'Naive'.
2. It Faces a *'zero-probability'* issue if you do not have a particular attribute value present in the training data, it will give a zero probability while calculating the posterior probability on the tuple that has that attribute value. This drawback can be handled using *'Laplacian Correction'* however.

Comparison

Naive Bayes is a linear classifier while K-NN is not; It tends to be faster when applied to big data. In comparison, k-nn is usually slower for large amounts of data, because of the calculations required for each new step in the process. If speed is important, choose Naive Bayes over K-NN.

Decision Tree

Decision tree is a binary tree which is used to classify the data into a particular class. We use the datasets provided to create training and testing sets. The training set is used to build the model which in this case is the decision tree. Once the model is trained we use the test set to check if the model is able to correctly predict the class of each tuple of the test set and calculate the accuracy of the model. We are using classification and regression trees(CART) decision tree algorithm.

Flow of Decision Tree Algorithm:

1. At each node in the binary tree we first check if all the tuples belong to the same class. If yes then the tuples are allocated to that node.
2. Otherwise for each column in the dataset we calculate the best split. The best split is calculated using the gini index. The gini index determines the best split based on the impurity of each split. The impurity of a split is the measure of homogeneity of the labels of the column in that split. Thus the split with the lowest impurity i.e higher number of tuples with same value is taken as the split for the current node and the data is divided based on that split.
3. The computation of best split is different in case of categorical and continuous values.
4. Best split calculation for categorical values:
    a. In case of categorical values such as "present" and "absent" as given in the dataset we first map them into numerical values. For example in this case we map "present" to 1 and "absent" to 0.
    b. Once all the values are in numerical form which makes it easier for computation we find all the possible combinations. For example if the categories where "cold", "sunny" and "humid" we map them to 0,1,2 and find the combinations as [0,1],[0,2],[1,2], this

becomes the split. We traverse each row in the dataset and check if the attribute value corresponds to the split accordingly we place it in one of the two sets i.e either it corresponds or it doesn't.

   c. Then we calculate the gini index for each set and select the set which minimizes the impurity. Then we repeat the process for each split and select the split with least impurity.

5. Best split calculation for continuous values:
   a. In case of continuous value we take continuous feature value of each row as a split and compute impurity using that split. The rows which has a value less than the split are placed in one split and the rest in other.
   b. Gini index is calculated for the set and select the set which minimizes the impurity. Then we repeat the process for each split and select the split with least impurity.
6. The above process is repeated until either all the records belong to the same class or when all records have similar attribute values.

Parameters

Tree Size:- we are taking the minimum size of the tree to be 1.
Depth size:- we are taking the minimum depth of tree as 3.

Pros of Decision Tree Algorithm:

1. No need to handle categorical values separately.
2. Easily combine with other ensemble techniques.
3. Pretty accurate when the dataset is small.

Cons of Decision Tree Algorithm:

1. Lack of robustness. Slight change in the values will result in a completely different decision tree.
2. Causes overfitting. Needs pre-processing and post-processing with techniques like pruning to increase the accuracy of the classifier.
3. May become complex when specifying large parameter values on large datasets.

Result

The average metrics for accuracy, precision, recall and f1-measure for 10 folds on both datasets are given below as follows:

The parameters are as follows: maxDepth=3 minSize=1

Max depth means maximum depth of the tree. Min Size means minimum 1 node in each branch required. Both are **stopping criteria**.

	Project3_dataset1.txt	Project3_dataset2.txt
Average Accuracy	**0.9195879120879121**	**0.6363224637681159**
Average Precision	**0.7947557282534399**	**0.47653171390013493**
Average Recall	**0.7978806378806379**	**0.489677398624767**
Average F Score	**0.7940047080944929**	**0.4763582537477937**

Comparison

1. We get a slightly higher accuracy for decision tree when compared to knn. This is because the decision tree gives a better result when there is a large set of categorical data which is present in our case.
2. Over-fitting problem can be dealt with using post-processing techniques like pruning where nodes of the tree are trimmed in a bottom up fashion.

Random Forest

Flow of Random Forest:

1. In random forest we compute a set of t classifiers for t subsets of training data.
2. In test phase each of the classifiers vote for a particular class for every row and the final class for that tuple is determined based on the majority votes. We are using bootstrap sampling with replacement in order to take the sample of data from training set each time we are creating a classifier. This allows for duplicate data to be present in the classifiers.
3. Hyperparameter t is chosen which determines the number of classifiers or trees to be constructed and each time a random selection with replacement of training data is done. This gives us t different trees or classifiers since each tree is created using different values.
4. We use another hyperparameter m.At each node we select m random features and find the split among the m features during construction of trees.A continuous feature can be used multiple times but categorical features can only be used once on the path from the root node to the current node. The construction of tree is explained in the decision tree algorithm.
5. After construction of all the classifiers or trees we use the same test data to determine the class and each classifier votes for a particular class. In the end we determine the class based on the majority votes.

Parameters

1. Hyper-parameter ‘t’ is the number of decision trees or classifiers in the random forest. A higher value of ‘t’ leads to lower variability in classification thereby reducing the overfitting.

2. Hyper-parameter ‘m’ is the number of features to be used while constructing the trees. ‘m’ should be a very small value(usually 20% of the overall features present in the dataset.

Pros of Random Forest

1. The problem of overfitting in decision tree algorithm is handled since here average of several trees is used for determining the final class.
2. It can handle large datasets since all the attributes are not used while creating decision tree.
3. Gives a better accuracy than decision tree.

Cons of Random Forest

1. As random forest uses several decision trees this increases the computational time making it slower than decision tree algorithm.
2. The result varies based on selection of number of trees and feature parameters making it difficult to find the optimal values for hyper-parameters.

The average metrics for accuracy, precision, recall and f1-measure for 10 folds on both datasets are given below as follows:

The tuning of hyper-parameters are as follows:- maxDepth=3 minSize=1
Max depth means maximum depth of the tree. Min Size means minimum 1 node in each branch required. Both are **stopping criteria**.

Tree Size = 1

	Project3_dataset1.txt	**Project3_dataset2.txt**
Average Accuracy	**0.943765664160401**	**0.6794326241134752**
Average Precision	**0.9276949651555924**	**0.5797137503019856**
Average Recall	**0.856603809054689**	**0.4437049062049062**
Average F Score	**0.8889866484323148**	**0.4661186294729938**

Tree Size = 5

	Project3_dataset1.txt	**Project3_dataset2.txt**
Average Accuracy	**0.9563317384370016**	**0.7152209492635024**

Average Precision	**0.9420112781954886**	**0.6233799533799533**
Average Recall	**0.8882599187988598**	**0.44980158730158737**
Average F Score	**0.9181883570715407**	**0.5111526375649798**

Comparison

1. We can see from the result that random forest gives a better accuracy than decision tree this is because the decision tree tend to overfit the model but since random forest uses a combination of several decision trees it solves the overfitting problem.
2. Since bootstrapping is used it reduces the variance of the prediction thus giving a higher accuracy as compared to that of decision tree.

Kaggle Competition

Here we have a real world dataset with training data set with 418x101 records, with 101 features and test data with 378x101 records. We performed various algorithms listed below and compared their individual results and combine them using majority voting methodology of ensemble learning. Parameters were tuned to give the best possible accuracy. All the algorithms are implemented using sklearn library.

Data Preprocessing

1. StandardScaler
   It transforms the data in such a manner that it has mean 0 and standard deviation as 1. In short, it standardizes the data. Standardization is useful for data which has negative values. It arranges the data in normal distribution. It is more **useful in classification than regression.**

2. Normalizer
   It squeezes the data between 0 and 1. It performs normalization. Due to the decreased range and magnitude, the gradients in the training process do not explode and you do not get higher values of loss. Is more useful in **regression than classification**.

<u>After testing each algorithm on Normalized and Scaled data, it was found StandardScaler give much better accuracy. Though, decision trees might give high accurcy for normalized for a particular seed value.</u>

Algorithm Implemented

1. Decision Tree
   Decision tree algorithm falls under the category of supervised learning. Decision tree uses the tree representation to solve the problem in which each leaf node corresponds to a class label and attributes are represented on the internal node of the tree.

Output for various values of depth on Training Data

max_depth	Accuracy	F1 Score
1	0.6904761904761905	0.5640509725016768
2	0.8095238095238095	0.7822128851540616
3	0.6904761904761905	0.6654641654641654
4	0.75	0.7669692593356716

Not used for majority voting as it have lowest accuracy.

2. K Nearest Neighbor

   In KNN we use the training records to predict class labels of unseen data i.e test data. In order to perform KNN algorithm we require a value of K to determine K nearest records known records that are close to the record whose class label is need to be identified

   Output for various values of depth on Training Data

n_neighbors	Accuracy	F1 Score
2	0.7976190476190477	0.8032116644500271
3	0.8333333333333334	0.8448587570621469
4	0.7261904761904762	0.7425086240620221
5	0.8095238095238095	0.8219435736677118

   Used for majority voting with n_neighbors = 3.

3. SVM

   The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

   Output for various values of depth on Training Data

Kernel	Accuracy	F1 Score
rbf	0.8095238095238095	0.7776610644257702
linear	0.8095238095238095	0.8193452380952382
poly	0.8452380952380952	0.8369745293466224

Poly was giving much better results as compared to others in Test Data as well. Degree and gamma are also pruned to 2 and 10.

4. Logistic Regression

   The logistic regression model predicts a dependent data variable(target) by analyzing the relationship between one or more existing independent variables(predictor).In this a line/curve is fitted to the data points, in such a manner that the differences between the distances of data points from the curve or line is minimized. Logistic regression calculates the probabilities of the data belonging to a particular class.

   Output for various values of depth on Training Data

c	Accuracy	F1 Score
0.1	0.8333333333333334	0.8240215924426451
0.15	0.7976190476190477	0.8067094450073172
0.2	0.8095238095238095	0.8029378531073446
0.3	0.7976190476190477	0.8059754725781229

   As the data set is small we use multinomial solver 'lbfgs', we set a random state and varied c value with iteration as 200 as at 100, it was not converging.

5. Random Forest

   Random forest consists of a large number of individual decision trees that operate as an ensemble. Each individual tree in the random forest spits out a class prediction. Each tree or classifier votes for a particular class and the resultant class is selected based on the majority of votes.

   Output for various values of depth on Training Data

n_estimators	Accuracy	F1 Score
10	0.8214285714285714	0.8093076719030916
15	0.8452380952380952	0.8310844577711143
20	0.7619047619047619	0.7148710230360801
25	0.7738095238095238	0.7427595950082411

   For n_estimators = 15,16,17,18 good f1 score and accuracy was observed, n_estimators = 16 was taken for test dataset.

6. AdaBoost Decision Tree

AdaBoost is a boosting ensemble model and works especially well with the decision tree. Boosting model's key is learning from previous mistakes, e.g. misclassification data points. Adaboost learns from the mistakes by increasing the weights of misclassified points. The tree with the highest weight will have more influence over the final classification.

Output for various values of depth on Training Data

n_estimators	Accuracy	F1 Score
5	0.7619047619047619	0.7522847522847523
6	0.7976190476190477	0.7744629014396457
7	0.8571428571428571	0.8408163265306123
8	0.7619047619047619	0.7444490992878089

We ran the model for both 6 and 7 in n_estimators, 7 was giving much better results for TestData.

Algorithm Chosen for Competition

Except decision tree, we used all of the other algorithm for competition as most have a good f1 measure. We used the method of majority voting from ensemble learning to determine the final class label for the test data set.

Efforts of Improvement

1. Data Preprocessing
   We both scaled and normalized the dataset to see which one performs better. Except the tree algorithm, in rest of all the algorithm Standard Scaling is much better than normalization. In tree based algorithm, output varies based on the seed value but for most seed values, Standard Scaling was better

2. Multiple Algorithm Implementation - Ensemble Learning
   Ensemble methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms alone. We used the same principle and applied on our classification algorithm and used majority voting method with 5 algorithm to get the final test label and tested the label on kaggle.

3. Parameter tuning
   The aim of parameter tuning is to find the optimum value for each parameter to improve the accuracy/f1 score of the model. To tune these parameters, we gained a good understanding of these meanings and their individual impact on model. Training data was split randomly and parameters range was chosen and final tuning was performed on the Test data set.

Due to scaling, outlier removal need not be done as it handles it, there were no missing value to correct as well.

Resources

- https://www.geeksforgeeks.org/basic-concept-classification-data-mining/
- https://www.geeksforgeeks.org/decision-tree-introduction-example/
- https://www.datacamp.com/community/tutorials
- https://medium.com/thalus-ai/performance-metrics-for-classification-problems-in-machine-learning-part-i-b085d432082b
- https://www.dataschool.io/simple-guide-to-confusion-matrix-terminology/
- https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7
- https://towardsdatascience.com/all-about-naive-bayes-8e13cef044cf
- https://www.geeksforgeeks.org/confusion-matrix-machine-learning/
- https://towardsdatascience.com/comparative-study-on-classic-machine-learning-algorithms-24f9ff6ab222
- https://medium.com/datadriveninvestor/k-nearest-neighbors-knn-7b4bd0128da7
- https://scikit-learn.org/