Data Mining Project – 3
Classification Algorithms

Sai Hari Charan Barla - 50336868

Hemant Koti – 50338178

Shravya Pentaparthi – 50337027

Performance Metrics

Classification Accuracy

Classification Accuracy is the ratio of the number of correct predictions to the total number of input samples.

F1 Score

F1 Score is the Harmonic Mean between precision and recall. It tells us how precise the classifier is (how many instances it classifies correctly).

High precision but lower recall, gives you an extremely accurate, but it then misses a large number of instances that are difficult to classify. The greater the F1 Score, the better is the performance of our model. Mathematically, it can be expressed as

$$F1 = 2 * \frac{1}{\frac{1}{precision} + \frac{1}{recall}}$$

F1 score tries to find the balance between precision and recall.

Precision

It is the number of correct positive results divided by the number of positive results predicted by the classifier.

$$Precision = \frac{TruePositives}{TruePositives + FalsePositives}$$

Recall

It is the number of correct positive results divided by the number of all samples that should have been identified as positive.

$$Precision = \frac{TruePositives}{TruePositives + FalseNegatives}$$

K-Nearest Neighbors Classification Algorithm

K-nearest neighbor is a lazy learning classification algorithm that classifies the data based upon the similarity between the test data record and all other records in the training data. It takes a single parameter – k – which is used to specify the number of closest neighbors that should be used to determine the label of the test record

Features

- *Supervised machine learning algorithm*: The target variable is known already.
- *Lazy algorithm*: It does not have a training step. Every data point will be used only at the time of prediction. Since there is no training step, the prediction step is costly.
- *Classification and Regression*: Used for both purposes.
- *Feature similarity*: It is the metric used to cluster the new point that will fall into.

Generic Algorithm

- Initially, the data is normalized.
- Then, the dataset is split into training and evaluation data based on the cross-validation technique.
- We have to decide the value for K.
- Once a new point comes, find the distance of the new point to every point in the training data.
- Then identify the K nearest neighbors to the new data point.
- From these k neighbors, the majority vote is taken and the label with the most votes is assigned as the label for that record.

K-Value

- There are no pre-defined statistical methods to find the most favorable K.
- But the choice of K will impact the results obtained from KNN drastically. Hence the value of K has to be chosen very carefully. Because Choosing a better value of K will improve the accuracy and precision of the results obtained.
- Choosing a lesser value of K might sometimes result in predicting with less accurate information and choosing a bigger value of K might result in including outliers and which will result in misclassification of data.
- So, it is very important to choose the right value for K.

Algorithm Implementation of our Code

1. Initially, we obtain the file name, the value of K, the value of K1 in K-fold from the user. The value of K is nothing but the number of nearest neighbors to be considered.
2. Once the file is obtained, it is passed as an input to the preprocess function and it is converted into an np array using np.asarray() function.
3. The data is preprocessed by converting the raw data into a matrix form using np.matrix() and ground truth is stored separately
4. While preprocessing, we normalize the data using the Min-Max Normalization technique. It linearly transforms x to y = (x-min)/max-min where min and max are the minima and maximum values in X, where X is the set of observed values of x.
5. Then we split the data according to a 10-fold cross-validation technique, where 90 % is used for training data and 10% is used for test data at each iteration.
6. KNN (X_train, X_test, Y_train, k) is called on split data.
    - For each X_test, the distance is calculated from every training data using the function np.sqrt(np.sum((X_train - t)**2,1)). It calculates the Euclidean distance.
    - Once the distances are calculated, the list is sorted.
    - From the sorted list top K training data indices are obtained. These are the top K nearest neighbors for that test data.
    - Once the indices are obtained, we check the training label and assign the max of the training label to the predicted test label.
    - The process is repeated for each of the test records in the K fold set.
7. Once the test labels are predicted, it is compared with the actual test labels.
8. The metrics are calculated by calling the function metrics (actual, predicted)
    - The true positives, true negatives, false positives, false negatives are obtained by comparing the actual test label and predicted test label. Then the metrics Accuracy, precision, recall, F1-measure are calculated by mentioning the above-mentioned formulas.
9. Now again we pick another dataset from the remaining 9 datasets. Each time the train and test data are split in a 90 to 10 percent ratio and the above steps are repeated for k1 times.
10. We continue until all the parts are covered, and the final metrics are calculated which are the mean of the values obtained at each step.

Final Results

1. For project3_dataset1.txt

```
enter the file name :project3_dataset1.txt
enter value of k in K-fold cross validation:10
enter value of k in K-NN :6
the mean values of metrics are :
Accuracy 95.35714285714286
Precision 94.43767521570724
Recall 93.72427693079868
F1-Measure 0.9401282931717715
```

```
enter the file name :project3_dataset1.txt
enter value of k in K-fold cross validation:10
enter value of k in K-NN :10
the mean values of metrics are :
Accuracy 95.53571428571429
Precision 94.47763979413595
Recall 93.86451065274596
F1-Measure 0.9409435214449445
```

2. For project3_dataset2.txt

```
enter the file name :project3_dataset2.txt
enter value of k in K-fold cross validation:10
enter value of k in K-NN :6
the mean values of metrics are :
Accuracy 65.21739130434784
Precision 43.6497549578045
Recall 50.686272061272064
F1-Measure 0.4588391303685421
```

```
enter the file name :project3_dataset2.txt
enter value of k in K-fold cross validation:10
enter value of k in K-NN :10
the mean values of metrics are :
Accuracy 66.30434782608697
Precision 46.06903145873734
Recall 51.999376354639516
F1-Measure 0.4836826271197795
```

Result Analysis

- Observation from the results is that KNN tends to perform better with the project3_dataset1.txt than project3_dataset2.txt.
- As can be seen from the results of project3_dataset1.txt, KNN works much better on continuous attributes.
- The performance is greatly affected when the dataset consists of categorical data as seen from the results of project3_dataset2.txt.

Advantages

- This algorithm is easy to implement.
- KNN works pretty well when the data attributes are continuous.
- It is pretty fast if the number of dimensions in the data is of reasonable size
- It is a very effective algorithm when the data set is large.
- There is no training phase involved.
- It learns complex models very easily.
- It is robust to noisy training data.
- Used in both classification and regression.

Disadvantages

- The number of neighbors i.e. 'k' can affect the classification of the records.
- Not very useful on data with categorical and continuous attributes mixed.
- The classification is greatly affected by noisy and unrelated data.
- It is computationally expensive as it searches the nearest neighbors for the new point at the prediction stage.
- False intuition is possible and sensitive to outliers which impacts accuracy in prediction.
- Sometimes, not clear which type of distance metric to be used.
- Huge memory requirement.

Decision Tree

A Decision tree is a flowchart like a tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label. The goal is to create a model that predicts the value of a target variable based on several input variables. At each node in the tree, we perform a split based on a particular attribute. The leaf nodes are the decision nodes that do not split. At each node we make an optimal decision to select a feature and a condition to perform the split. Below is the algorithm for the decision tree.

a) One-Hot Encode the input data to convert categorical attributes to binary.
b) At each node:
    a. We choose the best feature that gives the maximum information gain in this split and which has not been chosen before.
    b. We split the dataset into two sets based on the best feature and a condition.
    c. Then we pass one split dataset to the left child node and the other split dataset to the right child node.
c) Repeat step b) until a node does not meet any stopping criteria.

Stopping Criteria

We stop splitting at a node if any one of the following conditions is met.

- If a node has less than a specified number of observations.
- Depth of the node is more than the specified limit
- All the observations in the dataset have the same class label.
- If no attribute is left to split.

Entropy

The algorithm uses entropy to calculate the homogeneity of a sample. If the sample is completely homogeneous the entropy is zero and if the sample is equally divided it has an entropy of one.

$$E(S) = \sum_{i=1}^{c} - p_i \log_2 p_i$$

Information Gain

The information gain is based on the decrease in entropy after a dataset is split on an attribute. It helps in finding the most homogenous branches.

$$IG(D_p, f) = I(D_p) - \frac{N_{left}}{N} I(D_{left}) - \frac{N_{right}}{N} I(D_{right})$$

f: feature split on
D_p: dataset of the parent node
D_{left}: dataset of the left child node
D_{right}: dataset of the right child node
I: impurity criterion (Gini Index or Entropy)
N: total number of samples
N_{left}: number of samples at left child node
N_{right}: number of samples at right child node

Final Result

Following are the results on the entire dataset

Measure	Project3_dataset1.txt	Project3_dataset2.txt
Average Accuracy	91.07142857142857	65.21739130434783
Average Precision	86.66666666666667	46.666666666666664
Average Recall	96.29629629629629	46.666666666666664
Average F-Score	91.22807017543859	46.666666666666664

Following are the results for the 10-fold classification

Measure	Project3_dataset1.txt	Project3_dataset2.txt
Average Accuracy	93.21428571428571	60.86956521739131
Average Precision	88.24483738991746	42.475932478254464
Average Recall	92.79036519036518	45.338297571993216
Average F-Score	90.34605854722132	43.04466996975339

Advantages of Decision Tree

- Decision trees require less effort for data preparation during pre-processing.
- It does not require normalization and scaling of data.
- Missing values in the data do not affect the process of building a decision tree to any considerable extent.
- The model is very intuitive and easy to explain to technical teams as well as stakeholders.

Disadvantages of decision tree

- A small change in the data can cause a large change in the structure of the decision tree causing instability.
- Sometimes calculation can go far more complex compared to other algorithms.
- It involves a higher time to train the model.
- Training the decision tree is relatively expensive.
- The algorithm is inadequate for applying regression and predicting continuous values.

Random Forest

It is an extension of a decision tree. Random Forests uses multiple decision trees to work as an ensemble classifier. Every decision tree predicts a class label for a single observation and then finally the class label with the most votes is chosen as the model prediction. We use our implementation of a decision tree to build our random forest.

Pseudocode of Random Forest

1. Select T – number of trees to grow.
2. Choose m < Total number of features used to calculate the best split at each node.
3. For each Tree
    a. Choose a training sample of size N from the training dataset with replacement.
    b. For each node randomly choose the *m* features and calculate the best split.
    c. Fully Grown and Not Pruned i.e. each Tree is grown to the largest extent possible and there will be no pruning.
4. Use majority voting among all trees.

Algorithm

1. Convert all our categorical data into numerical values.
2. Create a subsample of our training dataset. Repeat this process for each tree.
3. Split the subsampled dataset to build our decision tree.
4. Select only a subset of the features instead of all the features while splitting the node.
5. Repeat the above steps for all the trees.
6. Take the maximum vote for each index in the dataset. This will be the final prediction of the random forest.

Final Result

Following are the results for the 10-fold classification

Number of Trees: 10

Number of features for the split: 5 (square root of all the features)

Measure	Project3_dataset1.txt	Project3_dataset2.txt
Average Accuracy	95.53571428571428	67.82608695652173
Average Precision	95.56376811594204	52.198762838468724
Average Recall	91.44440559440561	39.56666702862355
Average F-Score	93.27771260997068	44.34568071259481

Advantages

- Decorrelates trees.
- Can be used for both classification and regression.
- Incorporate more diversity and reduce variance.
- Prevents overfitting of data i.e. it won't overfit the model.
- It is used to handle large data set with higher dimensionality.
- Improves efficiency by searching among a subset of features rather than the complete set.

Disadvantages

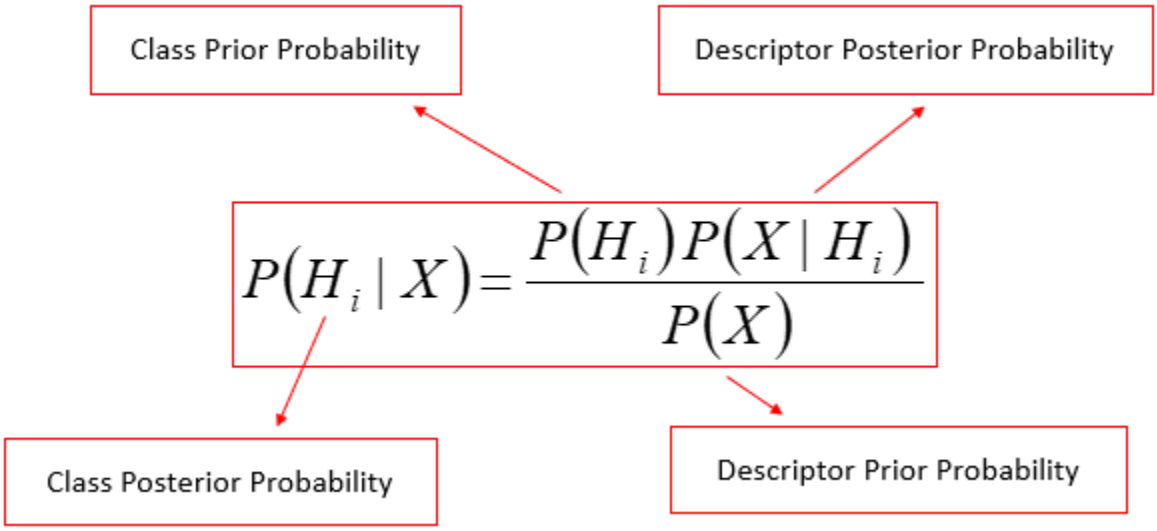
- Random Forests are not easy to visualize.
- They are much harder and time-consuming to construct than decision Trees. We avoid this problem by restricting the depth of each decision tree.
- Since there are too many classes, misclassification is also possible.
- It is a complex model.

Result Analysis

- From our implementation, we see that random forests give better accuracy than the decision tree because the random forest avoids over-fitting by decorrelating trees and reducing variance.
- We can see that model works better for datasets that have high non-linearity between different attributes.

Naïve Bayes

Naïve Bayes is a statistical classifier that is based upon Bayes Theorem. It assumes attribute independence and is efficient when compared to large databases. It is comparable in performance to decision trees. We predict the Naïve Bayes probability using the below formula:



Which states: The probability of a point being in a class is given by the product of the probability of a class consisting of that point times the probability of that class from the entire dataset over the prior probability of the tuple X.
Terms,

H is the hypothesis that a tuple X belongs to class C.

This algorithm assumes that all the attributes in the dataset are independent of each other. This assumption is called *class conditional independence*.

To find the class label of each test data record, we find it's the class posterior probability for each class and then compare these probabilities. Out of these probabilities, whichever class label has the maximum class posterior probability is assigned to this test data record. In our algorithm implementation, we're treating continuous and categorical attributes differently as explained in the sections below.

Continuous Attributes

When we come across continuous attributes, we find the mean and standard deviation for that attribute and use it to find Descriptor Posterior Probability for test data. We assume that test data follows a normal distribution and calculate P(X|H) using the gaussian distribution function.

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma_a} e^{-\frac{(x-\mu_a)^2}{2\sigma_a^2}}$$

where μ_a is the *sample mean*: $\mu_a = \frac{1}{|D_a|}\sum_{x \in D_a} x.a$

σ_a is the *sample standard deviation*, and

σ_a^2 the *sample variance*: $\sigma_a^2 = \frac{1}{|D_a|-1}\sum_{x \in D_a}(x.a - \mu_a)^2$

Categorical Attributes

When we encounter categorical attributes, we find that attribute's probability by counting the occurrences of each value in that attribute for that particular class and dividing it by the total number of records for that key.

Zero Probability

There can be instances where the test dataset has an attribute value that is not present in the training dataset which can result in getting a zero-descriptor posterior probability. To handle such instances, we will be using the Gaussian Distribution Function to determine descriptor posterior probabilities.

Results

Following are the results for the 10-fold classification

Measure	Project3_dataset1.txt	Project3_dataset2.txt
Average Accuracy	93.57142857142857	69.34782608695652
Average Precision	95.57124035384905	71.06774214702605
Average Recall	88.91630591630593	56.31809565638143
Average F-Score	91.85640874859546	61.14522554225023

```
enter file path :/content/sample_data/project3_dataset2.txt
enter value of k :10
the mean values of metrics are : 0.6934782608695652   0.7106774214702605   0.5631809565638143   0.6114522554225023
 accuracy : 0.7012987012987013  precision : 0.71875  recall : 0.5528846153846154  f1_measure : 0.625
CPU times: user 12.8 s, sys: 6.97 ms, total: 12.8 s
Wall time: 24.1 s
```

```
enter file path :/content/drive/MyDrive/Colab Notebooks/datamining project-3/project3_dataset1.txt
enter value of k :10
 accuracy : 0.9367311072056239  precision : 0.9481132075471698  recall : 0.8893805309734514  f1_measure : 0.9178082191780822
CPU times: user 37.9 s, sys: 3.98 ms, total: 37.9 s
Wall time: 1min 2s
```

Figures demonstrating the output for 10-fold classification.

Advantages of Bayes Theorem

- When the attributes are independent of each other then this algorithm gives apt results.
- It is easy to implement
- It handles both continuous and discrete data
- It is scalable with the number of data points and predictors
- It is fast and can be used to make real-time predictions
- It is not sensitive to irrelevant features

Disadvantages of Bayes Theorem

- Naive Bayes assumes that all predictors (or features) are independent, rarely happening in real life. This limits the applicability of this algorithm in real-world use cases.
- If a categorical variable has a category in the test dataset, which was not observed in the training dataset, then the model will assign a 0 (zero) probability and will be unable to make a prediction.

Result Analysis

- As can be seen, this algorithm performs well for continuous data as well as categorical data.
- The algorithm performs fairly efficiently even for large datasets.
- Naïve Bayes performs relatively better for dataset2 which consists of both numerical and categorical attributes.

Kaggle Competition

Objective

Apply various tricks on top of any classification algorithm discussed in class (including nearest neighbor, decision tree, Naïve Bayes, SVM, bagging, AdaBoost, random forests) and tune parameters using training data.

Algorithm

We tried out all the algorithms with different hyperparameters to find out the best performing algorithm. We used a train test split of 80 – 20% on the training data provided on all the algorithms.

We tried out KNN and Naïve Bayes which underfit the model and resulted in poor accuracy.

We tried the DecisionTree algorithm which overfits the training data and sometimes resulted in great accuracy. But the results were highly inconsistent with each random sample that is selected.

We observed that RandomForest gave us the best accuracy among all the algorithms. The results were fairly consistent with different samples. The accuracy we got ranged from 80 to 90%.

Hyperparameter Selection

The parameter selection in the above step was done manually. Once we finalized RandomForest we used **GridSearchCV** as shown in the notebook to select the best set of hyperparameters. The best hyperparameters we got for the model were.

bootstrap: true, n_estimators: 25, max_features: log2, max_depth: 12, criterion: gini

Results

Accuracy: 88.9740982

Precision: 87.903845

Recall: 80.9163030593

F1-Score: 80.8569087