Machine Learning

CS60050
Assignment-1 Report

Group 016

Ashwani Kumar Kamal (20CS10011)
Prerit Paliwal (20CS10046)

September 14, 2022

---

1 Problem Statement

1.1 Decision Tree Classifier

1. Split Dataset A into 80%-20% to form training and testing sets, respectively. Build a Decision Tree Classifier using **ID3 algorithm**. Train the classifier using **Information Gain (IG)** measure (no packages to be used for Decision Tree Classifier).
2. Repeat (1) for 10 random splits. Print the best test accuracy and the depth of that tree.
3. Perform **reduced error pruning** operation over the tree obtained in (2). Plot a graph showing the variation in test accuracy with varying depths. Print the pruned tree obtained in hierarchical fashion with the attributes clearly shown at each level.

1.2 Naïve Bayes Classifier

1. Randomly divide Dataset A into 80% for training and 20% for testing. Encode categorical variables using appropriate encoding method (in-built function allowed).
2. A feature value is considered as an outlier if its value is greater than mean + 3 x standard deviation ($\mu+3*\sigma$) sample having maximum such outlier features must be dropped. Print the final set of features formed. Normalise the features as required.
3. Train the Naïve Bayes Classifier using **10-fold cross validation** (no packages to be used for Naïve Bayes Classifier). Print the final accuracy.
4. Train the Naïve Bayes Classifier using **Laplace correction** on the same train and test split. Print the final accuracy.

2 Solution

2.1 Directory structure

- **DataCleaning.py** cleans the dataset, encodes it categorically and produces **cleanedData.csv**
- **DecisionTree.py** contains the code for implemented Decision Tree
- **Q1.py** is the solution file for first part of assignment
- **NaiveBayes.py** contains the code for implemented Naive Bayes classifier
- **Q2.py** is the solution file for second part of assignment

2.2 Data Cleaning

Dataframe manipulation has been handled by **pandas** library. In the final **cleanedData.csv**-

- *NaN* values have been replaced with the corresponding mode.
- Labels have been encoded categorically starting from 0.
- For **Age** feature, grouping has been done according to the ranges of size 10 (i.e. 0-9, 10-19, 20-29, etc ranges)
- For **Work_Experience** feature, values were initially in **float**, they are explicitly converted to **int** data type.

2.3 Decision Tree Classifier

2.3.1 Procedure

Main class and related functions are written in **DecisionTree.py**. A Node class is defined in this file to represent the Nodes of a Tree. The Node has the following data members:-

$attribute$: To tell which attribute of data we have to split on at this node.

$children$: A dictionary used to store children of the current Nodes based on what value they hold of the current attribute as the key.

$label$: every node is assigned a label as the target variable with the maximum frequency on the portion of data it receives.

$isleaf$: boolean variable to signify that is this node a leaf in the given tree.

1. Divide the dataset 10 times randomly in 80:20 ratio after shuffling to get the train and test dataset respectively with the help of **TrainTestSplit()** function. Then, further divided the train set into train set and validation set in a 70:30 ratio.
2. For each random spilt, first train a decision tree using ID3 algorithm. The **decisionTree(trianSet)** returns the root node of the decision tree after training.

3. The decision tree is generated by recursively calling a function **getNextNode(x,y)** which computes information gain for all the remaining attributes using the **getInfoGainList(x,y,attrs)** function and the greedily picks the one with max information gain and the is called recursively for its children. It stops when we are not left with any further attributes to divide on or if there is no information gained.

4. After training, the validation set is used to prune the tree to avoid overfitting on the training dataset. Starting from the current leafs, if removing them with the current label of node i.e. the target variable with maximum frequency, if this increases the accuracy we prune the subtree of that node.

2.3.2 Results

Final set of features formed are as follows-
{ Gender, Ever_Married, Age, Graduated, Profession, Work_Experience, Spending_Score, Family_Size, Var_1 }
Target variable- **Segmentation**

After 10 random splits -

- The accuracies obtained on Training, Validation and Test sets for the 10 random splits before and after pruning are mentioned in **output_Q1.txt**
- Best Test set accuracy after pruning: **42.65344079355239%**.
- The Decision Tree is printed in DecisionTree.txt clearly showing the depth level of the node and its children.
- The graph given below shows the accuracy variation with the depth of the tree. The values used in the graph are also mentioned in the last line of **output_Q1.txt**

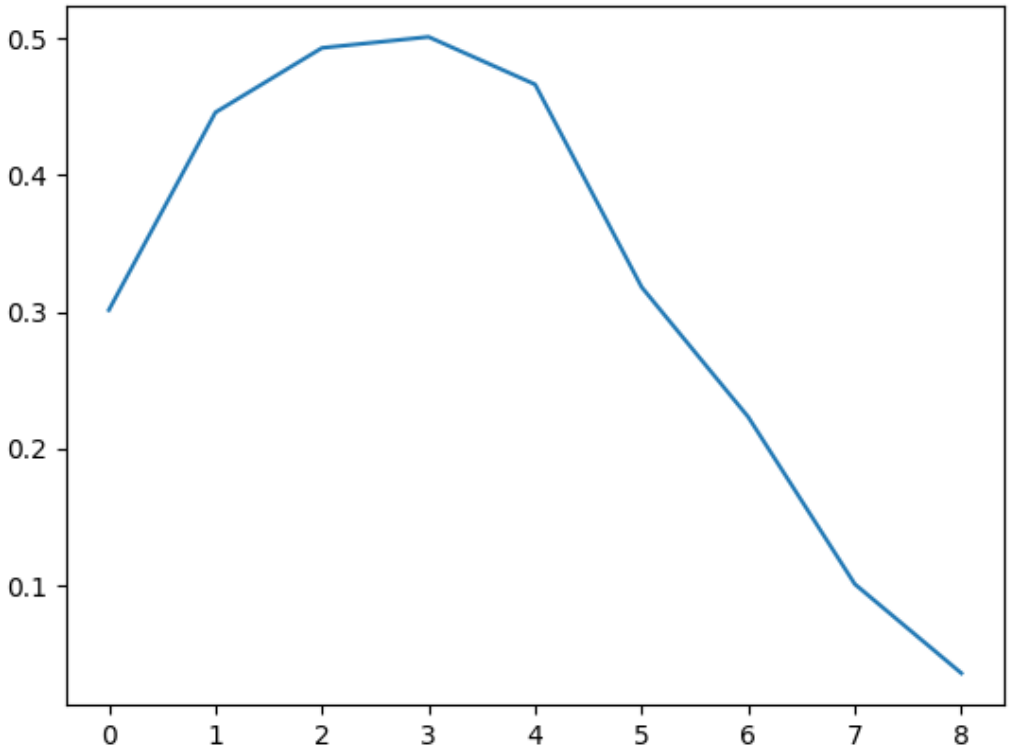

Figure 1: Accuracy VS Depth Plot

2.4 Naïve Bayes Classifier

2.4.1 Procedure

Main class and related functions are written in **NaiveBayes.py**. The NaiveBayes class has the following attributes- $train$: the training dataset, $test$: the testing dataset and n_folds: number of folds to be used in k-fold cross validation ($n\_folds = 10$).

1. Divide the dataset in 80:20 ratio after shuffling to get train and test datasets respectively. Set n_folds to 10. Instantiate the NaiveBayes class with these parameters. Remove the outliers as mentioned in problem statement after calculating mean and standard deviation for all features.

2. Further divide the training set into 10 subsets. For i^{th} iteration, set the i^{th} subset as the test set and remaining sets as training set. Related helper function: **cross_validation_split()**

3. Get the summary of the current training set, Essentially group the data according to target variables valid values and calculate the prior probabilities for each valid value of target. Calculate and multiply the posterior probabilities for each feature in this grouped.

Thus using the Bayes theorem, calculate the probabilities (assuming Gaussian distribution) for each valid target value. **Note:** In case of a 0 probability (happens when a particular feature value doesn't appear in current train set), ignore that probability as it will downgrade the accuracy and introduce error in the paradigm. Related helper function: **calculate_class_probabilities()**

4. Predict the final target value using the maximum final probability for each row in the current test set. Find the accuracy using actual and predicted values. Related helper function: **accuracy_metric()**

5. For all the 10 folds, find the average validation score. Get the train set which produces the highest accuracy.

6. Use the train set obtained in previous step, and find the predictions for original test set. Find the accuracy for this set (test accuracy)

7. In case of laplace correction, using the original train set, first calculate the prior and posterior probabilities for all the features and all valid values, multiply the corresponding the probabilities for predicting labels for original test set. Since there are no 0 probabilities remaining in this case, accuracy should be higher than that obtained in 10-fold cross validation.

2.4.2 Results

Final set of features formed are as follows-

{ Gender, Ever_Married, Age, Graduated, Profession, Work_Experience, Spending_Score, Family_Size, Var_1 }

Target variable- **Segmentation**

10 fold cross validation-

- The accuracies obtained on validation set on different iterations of 10-fold cross validation are- 49.61%, 49.15%, 48.53%, 49.30%, 45.27%, 49.77%, 48.06%, 46.82%, 46.20%, 51.01%.
- Average validation set accuracy: **48.37209302325581%**.
- The set having best score among these was then taken for testing against the original test set for which the accuracy came out to be **47.241165530068194%**.

Laplace Correction-

- Final test set accuracy improved and came out to be **51.51890886546807%**