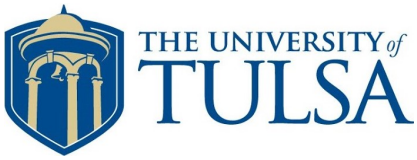

Deep Learning
CS-6863

Project 1

Instructor:
Mahdi Khodayar

Author:
Selim Karaoglu

February 16, 2022

This project is designed to implement Decision Tree and Naive Bayes classification methods to solve a specific problem; the banknote authenticity. The two classification techniques are trained and tested on banknote information dataset and the classification results are presented.

Introduction

In this project, two of the most commonly used classification methods, the Decision Tree and the Naive Bayes algorithms are implemented from scratch. Decision Tree and Naive Bayes methods are classification methods, therefore they can map the given attributes to some output. These classification methods serve for the same purpose, however they differ from each other with the way they make classifications. In machine learning terminology; Decision Tree is defined as a discriminative method while Naive Bayes is stated as a generative method. Both classifiers have their advantages and disadvantages, their success (classifications with high accuracies) depends on the nature of the dataset. Banknote dataset is a series of data that has 4 features (attributes) and a binary output; fake or real. This project is designed to utilize these classifiers to make classification on banknote dataset. Both classification methods trained on the banknote dataset and then resulting accuracies of the models are compared.

Classification Methods

In this section, background information about the Decision Tree and the Naive Bayes methods are presented. These methods are employed in data science studies to perform classification tasks on the given data. Even though both models are utilized for the same task, they differ from each other by the the mathematical background and the techniques that they used. By using these classification methods, the goal is to do a classification on the banknote dataset and train a classification model that can suggest how the datapoints can be labeled with the most accurate way. Each method is explained in detail to provide an insight about how they make classifications and how they differ from each other.

Decision Tree

Decision Tree is an upside-down tree that represent the decisions based on the given attributes. This method is implemented in machine learning researches with the purpose of making classifications based on the input data. Decision Trees naturally favor the data with categorical and conditional values. If the features are continuous, it can be converted to categorical data with an extra step. The upside-down structure of the tree allows to traverse the nodes of the tree to find the outcome owing to that each node consist of a feature that is split into more nodes down the tree structure. To build the Decision Tree, there is a necessity to compare and calculate the best split on features. To achieve this task and to decide how to split the tree, some splitting measure should be utilized.

With the consideration of more than one feature is affecting the decision making process, the priority should be evaluating the impact of each feature. With this evaluation, the most important or relevant feature takes its place on the top of the tree. As the next step, the same process is applied to second most relevant feature and this process is finished when all the attributes are contributed in the evaluation process. As this process continues, moving down the tree, the uncertainty and impurity decreases, hence the classification gets more accurate.

There are several methods to calculate the best splitting for the given attributes. Entropy is one of the splitting methods, it measures the uncertainty of a variable. This uncertainty of the variable "i" is calculated with: $E(S) = \sum_{i=1}^{n} -p_i \log_2 p_i$. Information Gain is another splitting method that is built on the Entropy method and aims to reduce the Entropy from the root to the leaves of the tree. Information Gain can be calculated with; $Gain(S) = E(S) - \sum_{v(A)} \frac{|S_v|}{|S|} E(S_v)$. Gini Index - Gini Impurity measures the probability of a variable being classified wrong and tries to minimize this to achieve better classification. On the calculation of Gini Index, the result varies between 0 and 1. Gini Index of 0 means that every attribute belong to a certain class or there are not multiple classes, when the Gini Index is 1, elements are randomly distributed to different classes. Gini Index can be obtained with: $Gini = 1 - \sum_{i=1}^{n} (p_i)^2$. Here the p_i refers to the probability of an element being classified to a certain class. When building the tree, the feature with the lowest Gini Index is assigned as the root note and the decision tree is built by repeating this step. Gini Index has more advantages comparing to Entropy and Information Gain, Gini Index is computationally less expensive comparing to other methods that employs logarithmic calculations. That's the reason that this project implements Gini Index as it's splitting measurement method.

Decision Tree implementation of this project adopts this algorithm:

Algorithm 1 Decision Tree

procedure NODE
returns if the input is leaf or not.
procedure DECISION TREE
function FIT
Initializes the root of tree with grow function.
function PREDICT
returns the predictions for the given data.
function GROW
grows the tree with utilizing most_common_label and _best_criteria functions.
function _BEST_CRITERIA
calculates the best splitting by utilizing gini_index function.
function GINI_INDEX
calculates the Gini Index for the given element.
function SPLIT
makes the splitting on the given point.
function TRAVERSE
moves from upside-down the tree to control the leaves and nodes.
function MOST_COMMON_LABEL
returns the most common label in the given data.

Naive Bayes

Naive Bayes is a machine learning algorithm, more specifically, it is a classification method. This means that Naive Bayes is used when the output variable is discrete. The underlying methods of the algorithm are based on the Bayes Theorem that states:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

To make it easier to understand, this equation can be written using the features (attributes, input values) and the targets (output values). In simpler terms; this function can be employed to solve for the probability of y with given features X as:

$$P(y|X) = \frac{P(X|y)P(y)}{P(X)}$$

The Naive assumption takes place here on the formula. Naive Bayes method works on cases where different feature values are independent from each other. Therefore the formula can be rewritten as:

$$P(X|y) = P(X_1|y)P(X_2|y)...P(X_n|y)$$

It is clear that the goal is to solve for y, in this circumstance P(X) is a constant value and it can be removed from the equation. In addition to that, by implementing the proportional nature of the $X_1, X_2, ..., X_n$ the equation can be presented as:

$$P(y|X) = P(y)\prod_{i=1}^{n} P(X_i|y)$$

Since the values across the datapoints can be calculated, the last step should be the finding the maximum of these calculated output values of y. The maximum can be obtained with the arg max function:

$$y = \arg\max_{y}[P(y)\prod_{i=1}^{n} P(X_i|y)]$$

With taking consideration on provided information about Naive Bayes, the classification method can be explained as; first, create a frequency table and then a ratio table so that the values for P(X) and P(y|X) can be calculated, furthermore, for a given set of input features X, compute the proportionality of P(y|X) for each class y. In this example, the implementation should be binary or simply True or False output. Now as the last important part of the Naive Bayes method, there could be different assumptions when calculating $P(X|y)$ like

Multinomial model, Bernoulli model and Gaussian model. This implementation adopts the Gaussian model with the assumption on the distribution of $P(X_i|y)$ follows a normal (Gaussian) distribution. Gaussian Naive Bayes model is based on the assumption of continuous values are sampled from a normal distribution, this assumption can be shown as:

$$P(X_i|y) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$$

Naive Bayes implementation for this work follows this algorithm:

Algorithm 2 Gaussian Naive Bayes

procedure NAIVEBAYES

function PRIOR

calculate prior probabilities P(y)

function STATS

calculates mean (μ) and variance (σ).

function DENSITY

calculates the probability from the Gaussian Distribution. $f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$

function POSTERIOR

calculates the posterior probability for each class and returns the class with the highest posterior probability.

function FIT

trains the model on the given dataset.

function PREDICT

returns the predictions for the given data.

Dataset

The dataset chosen for this project is the refined Banknote Authenticity Dataset provided by UCI. On the source of this dataset, authors defined the dataset as: "Data were extracted from images that were taken from genuine and forged banknote-like specimens. For digitization, an industrial camera usually used for print inspection was used. The final images have 400x 400 pixels. Due to the object lens and distance to the investigated object gray-scale pictures with a resolution of about 660 dpi were gained. Wavelet Transform tool were used to extract features from images. Data derived from UCI datasets"[1].

One of the most important aspects of this dataset is that all features are continuous. When the correlation heatmap is plotted, it can be seen if the features are correlated with each other. Figure 1 shows the correlation heatmap of the banknote dataset's attributes. As can be seen from the Figure 1, features of this dataset is not independent from each other.

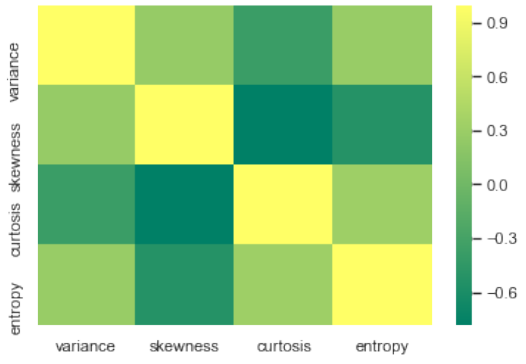

Figure 1: Correlation Heatmap of Dataset's Features

Target feature in the banknote dataset is defined as "class". It is a binary classification that value 0 responds to "fake" classification and value 1 responds to "real". Number of these classes can provide some insight about the dataset. There are 1372 datapoints in the dataset and 762 of these points are labeled as "fake" while remaining 610 points are labeled as "real". This distribution between classes are balanced, and balance in the outputs of the dataset makes classification problem easier to solve.

[1] Gediya V., et al. Bank Note Authentication UCI. https://www.kaggle.com/datasets/vivekgediya/banknote-authenticationcsv. 2021

Experiment

This section provides detailed information about the experimental process. First step of this process is to get the banknote dataset ready for the experiment. There is not much pre-processing needed for this dataset since this is a refined version of UCI banknote authenticity dataset. This means the empty values, wrong values etc. are cleared out of the dataset. There are still some actions needed to be done with the dataset like encoding the labels, separating the output values from features and splitting the dataset into train and test sets. After applying all these adjustments, the dataset is ready to be worked on.

First experiment

At this point, the classification methods are ready to train on the dataset. The Decision Tree and Naive Bates model are trained on the training dataset. This training dataset contains %70 of the complete dataset, the remanining %30 is the test dataset. After this training, the classification models are ready to make prediction on unseen data. This unseen data is the test dataset. When the models make predictions on new and unseen data, the predictions can be compared to the real values of the data and the accucacy of the prediction can be interpreted as the accuracy of the model. To calculate the accuracy model, a helper function called classification_report from sklearn[2] library is implemented. This function compares the predicted values and real values and returns accuracy scores with different measurement techniques such as precision, recall and f1-score. Decision Tree and Naive Bayes models are trained and tested, and the weighted average f1-score accuracy score for these classifications are %92 and %87 respectively. Figure 2 shows the complete output of the classification report. According to these accuracy scores, both Decision Tree and Gaussian Naive Bayes classification models can be utilized to classify this dataset with high accuracies, however; the Decision Tree model scores %92 accuracy and shows that it could be a better choice to solve this specific problem. As mentioned before, Naive Bayes method favors the discrete data, however, all attributes in the banknote dataset (variance, skewness, curtosis, entropy) are continuous valued attributes. This is the reason why Decision Tree has a higher accuracy scores and why Decision Tree model could be a better choice than Naive Bayes model for the banknote dataset.

```
Naive Bayes Classifier report:

              precision    recall  f1-score   support

           0       0.86      0.92      0.89       229
           1       0.89      0.81      0.85       183

   micro avg       0.87      0.87      0.87       412
   macro avg       0.87      0.87      0.87       412
weighted avg       0.87      0.87      0.87       412

Decision Tree Classifier report:

              precision    recall  f1-score   support

           0       0.91      0.96      0.93       229
           1       0.94      0.89      0.91       183

   micro avg       0.92      0.92      0.92       412
   macro avg       0.93      0.92      0.92       412
weighted avg       0.93      0.92      0.92       412
```

Figure 2: Classification Report for Naive Bayes and Decision Tree models on banknote dataset

Different Dataset Splits

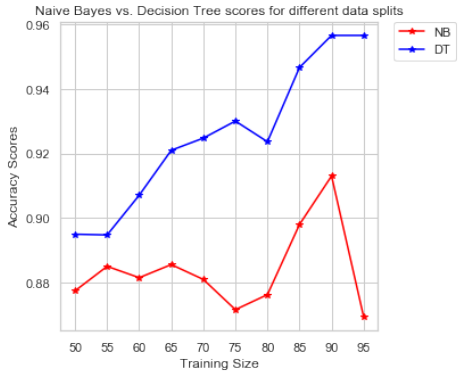

Figure 3: Classification Report for Naive Bayes and Decision Tree models on banknote dataset

The accuracies of the classification models are dependent on some factors. Above, the significance of the attributes has shown. One other criteria that may effect the accuracy of the classification models is the dataset size. As a part of the experiment, the training and test splits are varied and both classification methods are repeatedly trained and tested with these different dataset splits. Figure 3 shows the accuracy scores for Decision Tree model (blue) and Naive Bayes model (red). As expected; both models have increasing accuracy with the increasing data size. Nevertheless this accuracy change occurs only in %0.07 range. Both models have reasonable accuracies on all training-test splits. Figure 3 shows also that Decision Tree performed better than Naive Bayes on all different dataset splits.

[2]https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

Maximum Depth of the Decision Tree

As a final step of this experiment, the Decision Tree model is trained on the same banknote dataset with different maximum depth value. Maximum depth of a tree is a rule to build a Decision Tree with a certain height to prevent overfitting. But deciding the height of the tree depends on the nature of the dataset. On this step, the Decision Tree model trained on the dataset with varied maximum depth values and the accuracy scores for each model are compared to decide the best maximum depth value decision. Figure 4 shows the accuracy scores for Decision Tree model with different maximum depth values, as shown on the figure, the first 5 levels of depth has a high impact on the models the accuracy, however, after the 5th level, the accuracy of the model doesn't show significant increase, although the accuracy scores reach as high as %98 on training.

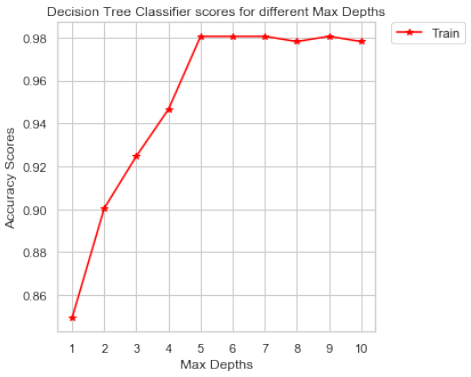

Figure 4: Decision Tree models with different Maximum Depth

Conclusion

Decision Tree model and Naive Bayes model are implemented and utilized to solve banknote authenticity problem. First the classification models are constructed, then the dataset is pre-processed and splitted for training and testing. Both models performed reasonably good on this dataset, Decision Tree model outperformed the Naive Bayes with %6 higher accuracy. This is due to the nature of the banknote dataset; while Naive Bayes model performs better on the attributes with discrete values, all of the attributes of the banknote dataset are continuous. This problem might be overtaken by categorizing the continuous values, but this is currently out of scope for this project and might be implemented on future work.