University of East London

Msc in Data Science

Advanced Decision Making: Predictive Analytics & Machine Learning
DS7003

Comparing categories of machine learning algorithms through decision-making exercises on selected datasets.

Student ID:
2121807

June 2021

Abstract

This paper aims to explore the process of decision making using Machine Learning algorithms and methodologies. The development is divided into two parts. Firstly, we will compare tree-based algorithms with conditional probability algorithms. Secondly, we will train and compare Machine Learning algorithms to solve a prediction problem of predicting the scribes of the Avalone Bible based on page layout features. The Avila Dataset with 10,430 training samples and 10,437 test samples will be used for this purpose, and Python will be the programming language utilized. This is a supervised learning problem since we already have information on the passages of the Bible written by each scribe. The prediction problem will be approached as a classification problem, and the results of the different algorithms will be analyzed and compared.

Table of contents

Module 1

Essay

This module is devoted to the presentation and comparison of two broad categories of decision algorithms, namely those based on decision trees and those based on conditional probabilities. Each class will be discussed in detail, including how they function, their primary advantages and limitations, and the applications that motivate their use. In addition, a comparison between the two categories of algorithms will be made, highlighting their similarities and differences.

1.1 Decision Tree Algorithms

Decision Trees are a type of Supervised Learning algorithm that can be used for both classification and regression problems. The concept of decision trees was first introduced in 1959 by William Belson in his research on the Principles of Biological Classification. Decision tree algorithms use tree-like structures to make predictions based on observations about an object. Starting from the root, the algorithm compares the value of the tree root with the values of the sample parameters, and based on this comparison, it chooses which path to follow and which node to take next. This process is repeated until the algorithm reaches a final node or leaf, where a decision on the category or value of the observation is made.

To provide a better understanding of decision tree algorithms, Figure 1.1 shows an example where each observation is a Titanic passenger, and the algorithm uses attributes such as gender, age, and sibsp to predict whether the passenger survived or not. In the tree, the root and intermediate nodes compare available attributes with fixed values, while the final nodes or leaves are used to make the decision. The fixed values of the intermediate nodes are determined by the learning process that the tree goes through.

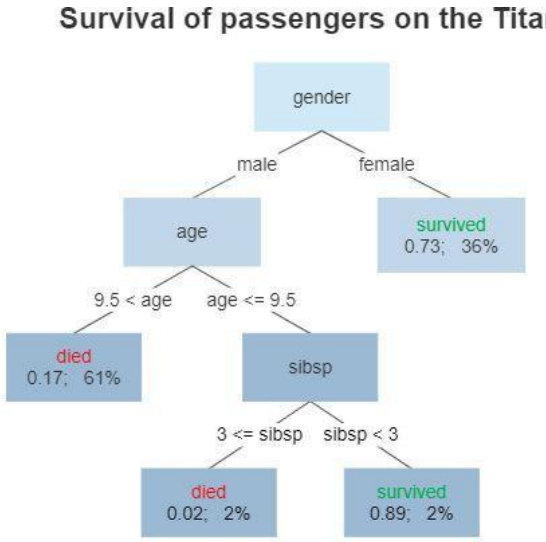

Figure 1.1: Decision Tree example

Constructing a decision tree involves strategically dividing the feature space, which is achieved through a learning process. Decision tree algorithms determine when to split a node into multiple subnodes, with the goal of creating more homogeneous nodes. This leads to nodes with similar samples in relation to the target variable. The algorithms assess all possible ways to split a node and select the one that maximizes performance. The subsequent section will delve into three of the most widely used decision tree algorithms: ID3, C4.5, and Random Forests.

1.1.1 Algorithm ID3

The ID3 algorithm utilizes a top-down, greedy search approach to build decision trees by exploring the potential branches of the feature space without revisiting previous choices. The algorithm proceeds through the following steps:

1. The initial feature set S is set as the root of the tree.
2. In each iteration, the algorithm assesses the potential splits for each feature in S and calculates both the Entropy and Information Gain (IG).
3. The feature with the lowest Entropy or the highest IG is then selected.
4. The selected feature partitions the set S into subsets.
5. The algorithm repeats the process on each subset, excluding the previously selected features.

ID3's ability to discover the optimal solution is not guaranteed, as it may become restricted to local minima due to its greedy approach in each iteration. The algorithm's lack of backtracking is another limitation, as it could potentially yield better results but at a higher computational cost. Furthermore, ID3 is considerably limited and slow when the target variable is continuous, as the extensive branch choices can hinder its efficiency. Overfitting to the training data is another issue that can significantly reduce its ability to generalize to new data.
ID3 represents a relatively simplified version of decision tree algorithms, and it may have practical applications in problems where training time is a priority, such as online learning. It is also a valuable tool for educational purposes in understanding

how these algorithms function.

1.1.2 Algorithm C4.5

The C4.5 algorithm [4] is an extension of the ID3 algorithm and is capable of generating decision trees that can be used for both classification and regression tasks. The generation process is similar to ID3, and the pseudocode is not presented here since it closely resembles that of the previous algorithm.

C4.5 provides several enhancements over ID3. One such improvement is its ability to manage both continuous and discrete variables. C4.5 establishes a threshold and then splits the list of values into two groups: those exceeding the threshold and those falling below it. The algorithm is also capable of handling data with missing values by merely ignoring them during the process of computing Entropy and IG. Finally, C4.5 features a significant improvement in its use of backtracking to prune the decision trees by eliminating subtrees with little value and replacing them with leaf nodes. This approach reduces overfitting of the estimator to the training data.

1.1.3 Random Forests

In this chapter, we will discuss the Random Forests algorithm [5] [6], which is a set learning methodology. The fundamental idea behind this algorithm is to train a set (forest) of estimators for a problem instead of just one decision tree. Random Forests utilize a technique called bagging, which is used to create a set of trees that use multiple training data sets. These datasets are generated from the original dataset by randomly selecting samples through re-sampling. Therefore, a unique random training dataset is created for each tree in the forest as a subset of the original set. The bagging method not only selects random samples for each new subset but also randomly selects the features to be used in each set. As a result, each Random Forests tree is trained on a different dataset with different attributes.

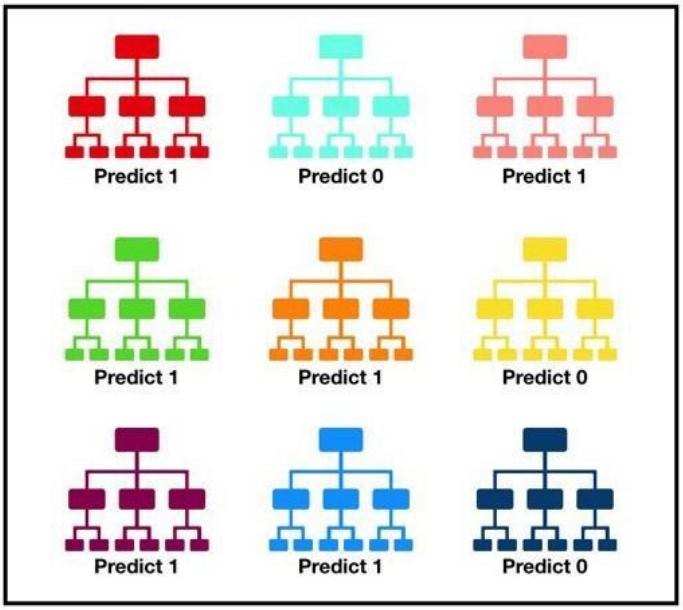

Figure 1.2: Random Forest example

Random Forests use trained trees in their prediction process. For a sample whose output value is to be predicted, the sample is passed through all the trees, and each tree generates an estimate (vote) for that sample. The predictions from all the trees are then combined to

produce a final prediction, which can be made by different methodologies such as majority voting or averaging individual decisions.

Random Forests are very powerful estimators due to the wisdom of the set they employ. They consist of a large number of relatively uncorrelated decision trees, with the feature of uncorrelatedness being key to their success. Each tree protects the other from its individual error, resulting in excellent generalization. Compared to other machine learning algorithms, Random Forests produce powerful estimators for handling both linear and non-linear relationships, as well as truncated values.

Random Forests is a methodology that utilizes multiple trees to create powerful estimators that often outperform simple decision trees. This algorithm trains tens or hundreds of trees to solve a problem, resulting in better estimator performance, but it also has some drawbacks. For instance, the easy interpretability of the resulting decision trees is lost due to the large number of trees constructed. As a result, the programmer cannot easily track the actions taken by the Random Forest and may only see it as a black box. Another limitation of Random Forests occurs when the dataset includes categorical variables, which may result in lower performance than a single decision tree, according to [7]. Finally, it's worth mentioning that these algorithms create complex models that can be computationally costly, particularly on large datasets, when compared to decision trees.

1.2 Conditional probability algorithms

In the field of machine learning, a probabilistic estimator refers to an estimator that can predict a probabilistic distribution of a set of classes or a continuous output variable given an input observation. These estimators are based on the principle of conditional probability, which is a fundamental concept in the field of probability. Conditional probability is defined as the probability of an event A (dependent variable) occurring given another event B (independent variable) and is represented by P(A|B). Hence, algorithms that exploit the conditional dependence of the output variable on the input variables are categorized under this section.

This paper will focus on three main categories of machine learning algorithms based on conditional probabilities, namely the Linear Regression algorithm, the Accounting Regression algorithm, and the Able-Base algorithm. Although there are several other machine learning algorithms that use probabilistic theory and the conditional probability tool, this paper will not cover all of them.

1.2.1 Linear Regression

Linear Regression is a type of estimator that assumes a linear relationship between the input and output variables. It is based on the equation $y = Wx + b$, where W and b are the coefficient vectors that need to be estimated. The ideas for linear regression date back to the 19th century and have been studied extensively since then. There are several methods for estimating the parameters of linear regression, including Normal Least Squares, Maximum Likelihood Estimation, Ridge Regression, and Slope Descent. In this paper, we are particularly interested in the Bayesian Linear Regressor, which utilizes conditional probabilities. This approach assumes that the coefficients W and b are random variables with a prior probability distribution. Rather than producing a specific estimate for the regression coefficients, this algorithm generates a posterior probability distribution that describes the uncertainty associated with the output variable. This distribution can then be used to calculate the optimal coefficients.

Linear regression is a widely recognized machine learning algorithm that has numerous applications in fields such as biology and social and behavioral sciences, effectively describing the relationships between variables. It performs particularly well when the data has linear relationships or dependencies, and is easy to apply, train, and interpret. However, it is not without its limitations. One of the drawbacks is the need to make the often incorrect assumption that there is a linear dependence between the dependent and independent variables. Another drawback is that models produced by this method

are prone to noise and overfitting. Moreover, linear regression models are relatively simple, making them difficult to use for complex problems with large datasets..

1.2.2 Logistic Regression

An accounting regression is a statistical technique that models the probability of a specific event occurring. While logistic regression is typically used in classification problems, there are also extensions that can be applied to regression problems. The most common use case is binary classification, where the algorithm predicts the likelihood of an event occurring with a probability close to one or not occurring with a probability close to zero. Similar to linear regression, the output y is determined by input x with y = Wx + b, but instead of a linear function, a sigmoid function is used. The sigmoid function, which resembles the shape shown in Figure 1.3, pushes positive values towards one and negative values towards zero. In the case of multiple output classes, the sigmoid function is extended to the softmax function. The learning process for logistic regression involves finding the optimal parameters W, b using techniques such as Probability Maximization and Least Squares.

Logistic regression has the advantage of being easy to apply and interpret, as well as being fast and efficient for small datasets with a small feature space. Another advantage is the ability to predict both the correlation between input and output variables and the direction of the correlation. However, the algorithm's simplicity limits its performance with large datasets. Additionally, logistic regression cannot solve nonlinear problems because it develops a linear decision surface similar to linear regression. A significant difference from linear regression is that the independent variables in logistic regression are linearly correlated with the logarithm of the probabilities, whereas in linear regression, the dependent variables are linearly correlated with the independent variables.

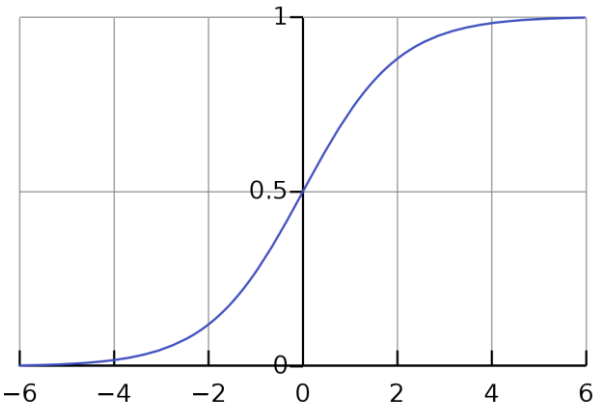

Figure 1.3: Sigmoid function

Kramer [9] credits Pierre François Verhulst in the 1830s as the originator of the concepts that underlie logistic regression, which serve as a fundamental building block for more advanced machine learning algorithms such as the perceptron [10] and artificial neural networks [11]. Overall, the logistic regression algorithm is a highly effective tool for modeling probabilities and has numerous applications in machine learning and beyond.

1.2.3 Naive Bayes

Naive Bayesian classifiers [12] are a category of classifiers used in statistics and machine learning, based on Bayes' theorem. According to this theorem, the posterior probability equals P(A|B) = P(B|A)P(A)/P(B), where P(A|B) is known as the conditional probability, P(B|A) is the likelihood, and P(A) is the prior probability. These classifiers are called "naive" because they assume that the features of the problem are independent of each other. The Bayesian estimators are trained using the Maximum

Likelihood method to compute the probability distributions required to calculate the posterior probability, which represents the probability of a sample belonging to class A given its characteristics B.

Although the assumption of independence between features is often unstable, naive Bayesian classifiers possess some characteristics that make them extremely useful in practical applications. In particular, the decoupling of the conditional probability distributions of each class enables each distribution to be computed separately and independently. Furthermore, these classifiers have a duality with logistic regression classifiers when the input variables are categorical, creating a pair that solves the same problem with the same probabilistic model fitting, since logistic regression optimizes the conditional posterior probability while Bayesian optimizes the likelihood.

The assumption of independence mentioned above is a crucial point of the algorithm and has both negative and positive implications. The positive aspect is that with this assumption, the algorithm becomes very fast, making it ideal in cases where speed is more important than accuracy. However, the negative side is that this assumption often contradicts the nature of the problems the algorithm is trying to solve, resulting in a decrease in the accuracy of the prediction. Another advantage of the algorithm, which is not found in algorithms of the same class, is that it is also highly efficient in problems with large, multidimensional data.

1.3 Comparison of Algorithms and Conclusions

Decision trees are estimators that offer flexibility, robustness, and ease of interpretation and debugging. They are equally powerful in both classification and regression problems. Additionally, the random forest algorithm can combine several decision trees to create even more accurate and robust classifiers at a higher cost. However, decision tree classifiers are not universally applicable and successful. Therefore, we compare decision trees with three categories of algorithms that exploit conditional probabilities.

Firstly, decision trees are compared to the linear regression algorithm. Decision trees can support non-linearity, but linear regression is an ideal solution when the solution is governed by linearity. Linear regression also performs better when the available data is small in volume but with many characteristics, as the linear equation can model the relationship between input and output much better.

Compared to logistic regression, decision trees are considerably superior. Decision trees can handle missing values and outliers, unlike logistic regression in the learning process. Additionally, decision trees can handle non-linearity. However, logistic regression has an important advantage over decision trees: the ability to calculate the degree and direction of the correlation that each characteristic has with the output.

Naive Bayesian classifiers are effective and accurate models that differ from decision trees in several ways. Firstly, Bayesian models are generative, which means they produce desired distributions. On the other hand, decision trees are discriminative models, as they use feature values to create branching and splitting. Secondly, decision trees automatically select the most significant features for learning, whereas this has to be done manually by the programmer with Bayesian models. Additionally, Bayesian models generate probabilities, which can make interpreting the trained model challenging. Bayesian models are better suited for small data volumes, whereas decision trees require a significant amount of data to be trained efficiently. However, decision trees are more prone to overfitting than Bayesian models, especially without the application of pruning. Nonetheless, selecting an algorithm should be done thoughtfully, taking into account the specific characteristics of the problem at hand. A practical comparison of the models' performance is an effective method for selecting the most appropriate algorithm. This will be discussed further in the following module.
.

Module 2

Decision Making Project - Correlation of the Avila Bible Standards with the Corresponding Authors Using Machine Learning Techniques

2.1 General

2.1.1 Abstract

This chapter will focus on a machine learning task where we attempt to predict the authorship of different parts of the Bible among 12 potential copyists-authors. The basis of our predictions will be writing style characteristics. We will approach this problem as a classification task, and we will compare the performance of two classification models: a Naive Bayes Classifier and a Random Forest Classifier. Finally, we will present a comparison of the two models we have trained and showcase their respective results.

2.1.2 Introduction

The aim of this paper is to predict the specific author-antichrist responsible for each section of the Avila Bible using the Avila Dataset from the UCI Machine Learning Repository. The dataset was originally collected by C. De Stefano, F. Fontanella, M. Maniaci, A. Scotto di Freca in their study titled "Reliable author identification in medieval manuscripts through page layout features: the Avila Bible case" [13]. In their research, the authors developed a writer identification system by extracting page layout features from images of medieval texts in the Avila Bible. Several other studies have utilized this dataset over time, including those by Krishna Kumar Sharma [14] on pattern clustering using φasmat k Nearest Neighbors, Ishwar Baidari [15] on diverse online boosting, and Robledano Arillo [16] on testing the quality of digital text representations.

The Avila Dataset consists of 800 images of the Avila Bible, a XII century Latin text that was copied by at least nine different scribes during the third decade of the XII century in Italy. It was then transported to Spain, where local scribes completed the text and decoration. In a third phase, during the fifteenth century, the Avila Bible was further adapted by another scribe to meet new linguistic and functional requirements. The collaboration of numerous scribes in creating the Avila Bible makes it particularly interesting to attempt to identify and record the sections that each of them wrote. In Figure 2.1, some of the original book pages are shown, providing a better understanding of its current form..

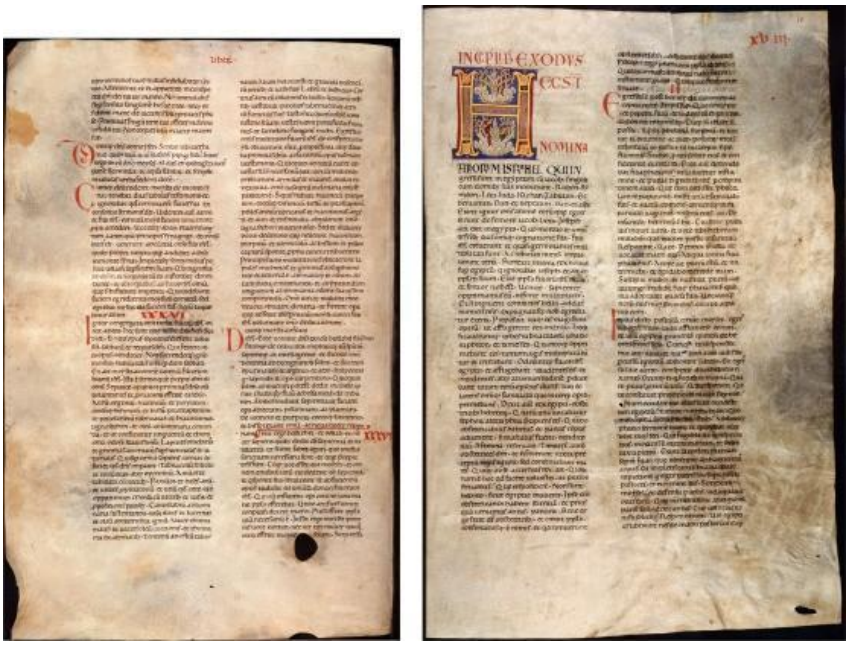

Figure 2.1: Avila Bible

2.2 Dataset

2.2.1 Dataset characteristics

The data available for analysis is sourced from the Avila Dataset and consists of 10 features that are classified into three categories according to the authors [13]. The first set of features is related to the geometric properties of the entire page image and comprises of attributes such as upper margin, lower margin and intercolumnar distance. While these attributes may not be very distinct for individual scribes, they can be helpful in highlighting topological and chronological differences. The second set of features pertains to the analysis of each column of the written area and includes attributes such as row number and exploitation coefficient, which measures the extent to which a column is filled with ink. The third set of features describes the distribution of writing for each line of text and comprises of attributes such as weight, peaks number, interlinear spacing, modular ratio, and modular ratio-interlinear spacing. The weight attribute measures the degree to which a line is filled with ink, while the modular ratio is the ratio of the horizontal size of a character to its vertical size.

Name	Data Type	Description
Intercolumnar Distance	continuous	Distance between columns
Upper Margin	continuous	Upper margin of the page
Lower Margin	continuous	Lower margin of the page
Exploitation	continuous	Exploitation coefficient (how much the column is filled with ink)
Row Number	continuous	Number of rows in each cloumn
Madular Ratio	continuous	Ratio between the horizontal and vertical size of a character
Interlinear Spacing	continuous	Distance between rows)
Weight	continuous	Weight coefficient (how much each row is filled with ink)

Peak Number	continuous	Mean number of peaks per page
Madular Ratio-Interlinear Spacing	continuous	Combination of Madular Ratio and Interlinear Spacing
Class	nominal	Coresponding scribe

Table 2.1: Dataset characteristics

2.2.2 Dataset description

The data used in this study are sourced from the Avila Dataset [13]. The authors of the dataset measured 800 images of various pages from the Avila Bible, and the resulting dataset has two sub-sets: a training set comprising 10,430 samples, and a control set consisting of 10,437 samples. Neither of these sets has any missing values. Each sample is described by 10 continuous variables, and the target variable is a categorical variable with 12 possible values {A, B, C, D, E, F, G, H, I, W, X, Y}, which correspond to the 12 scribes of the Bible. The aim of the study is to predict the values of the target variable. Table 2.1 shows the target variable and its values. Table 2.2 displays some statistics for the measurements taken from the available data, which have been normalized by the authors using the z-normalization methodology.

	interc. dis.	up. marg.	lo. marg.	exp.	modular r.	interl. sp.	rows	weight	peaks	mr-is
Min	-3.498	-2.426	-3.210	-5.440	-4.922	-7.450	-11.935	-4.247	-5.486	-6.719
Max	11.819	386.0	50.0	3.987	1.066	53.0	83.0	13.173	44.0	4.671
Mean	0.0	0.033	0.0	-0.002	0.006	0.013	0.005	0.01	0.012	0.0
STD	0.991	3.920	1.120	1.008	0.992	1.126	1.313	1.003	1.087	1.007

Table 2.2: Dataset characteristics

Figure 2.2 displays the distributions of both the characteristics and the meta-variables. The variables Intercolumnar Distance, Exploitation, Weight, and Modular Ratio-Inerlinear Spacing exhibit distributions that closely resemble the normal distribution. Moreover, the variables Upper Margin, Lower Margin, and Interlinear Spacing follow a nearly uniform distribution. Figure 2.3 illustrates the distribution for the data variable pertaining to the output categories. Unfortunately, it is apparent that the data is not evenly distributed across the 12 categories, which is not ideal. This uneven distribution necessitates a distinct approach to the model training process. It should be noted that all of the previous information pertains solely to the training dataset.

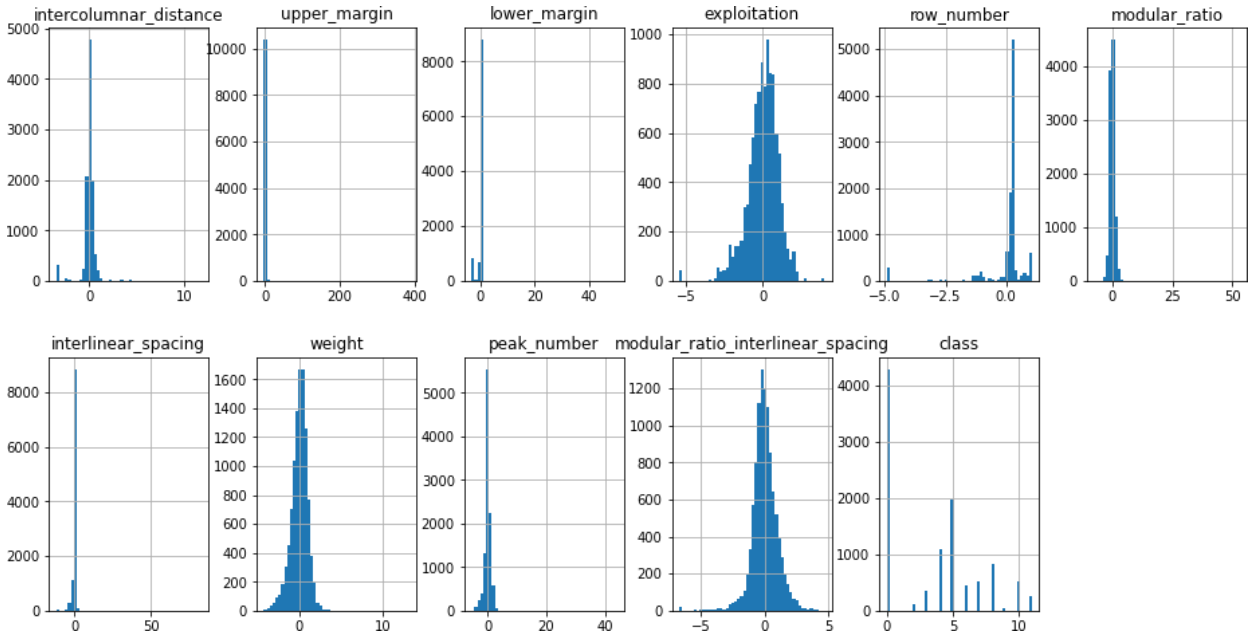

Figure 2.2: Attribute distributions

2.2.3 Processing and Study of the Dataset

It is worth noting that the target variable (class) has been modified such that the class values, which originally consisted of uppercase letters of the English alphabet, have been replaced with consecutive integers ranging from 0 to 11. This conversion is depicted in Figure 2.3.

Figure 2.4 showcases the correlation between two characteristics of the dataset. This type of analysis, known as bivariate analysis, is crucial in the process of data analysis because it provides a clear picture of how each attribute is influenced by the presence of others. It also aids in identifying important characteristics. In this study, we observe a linear correlation between the Modular Ratio and Interlinear Spacing variables with the Modular Ratio-Interlinear Spacing variable. However, none of the other relationships between the various variables exhibit any correlation of particular significance or usefulness.

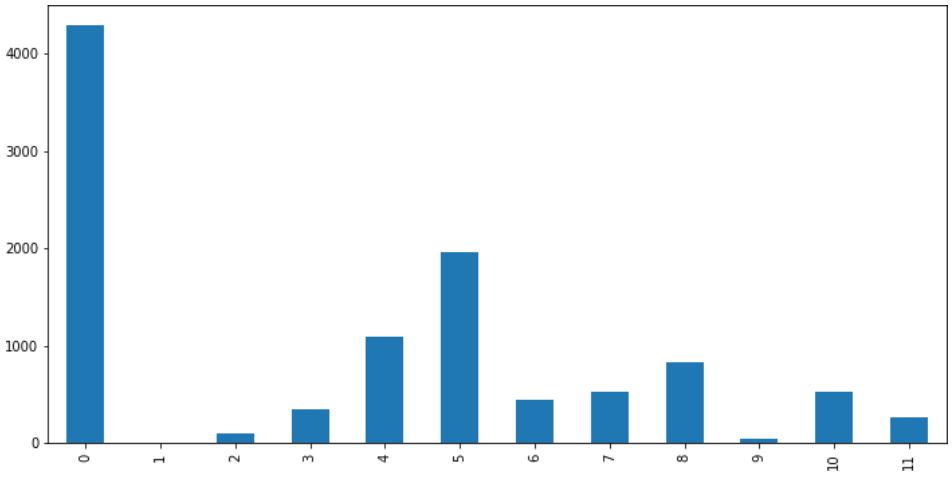

Figure 2.3: Distribution of categories (training set)

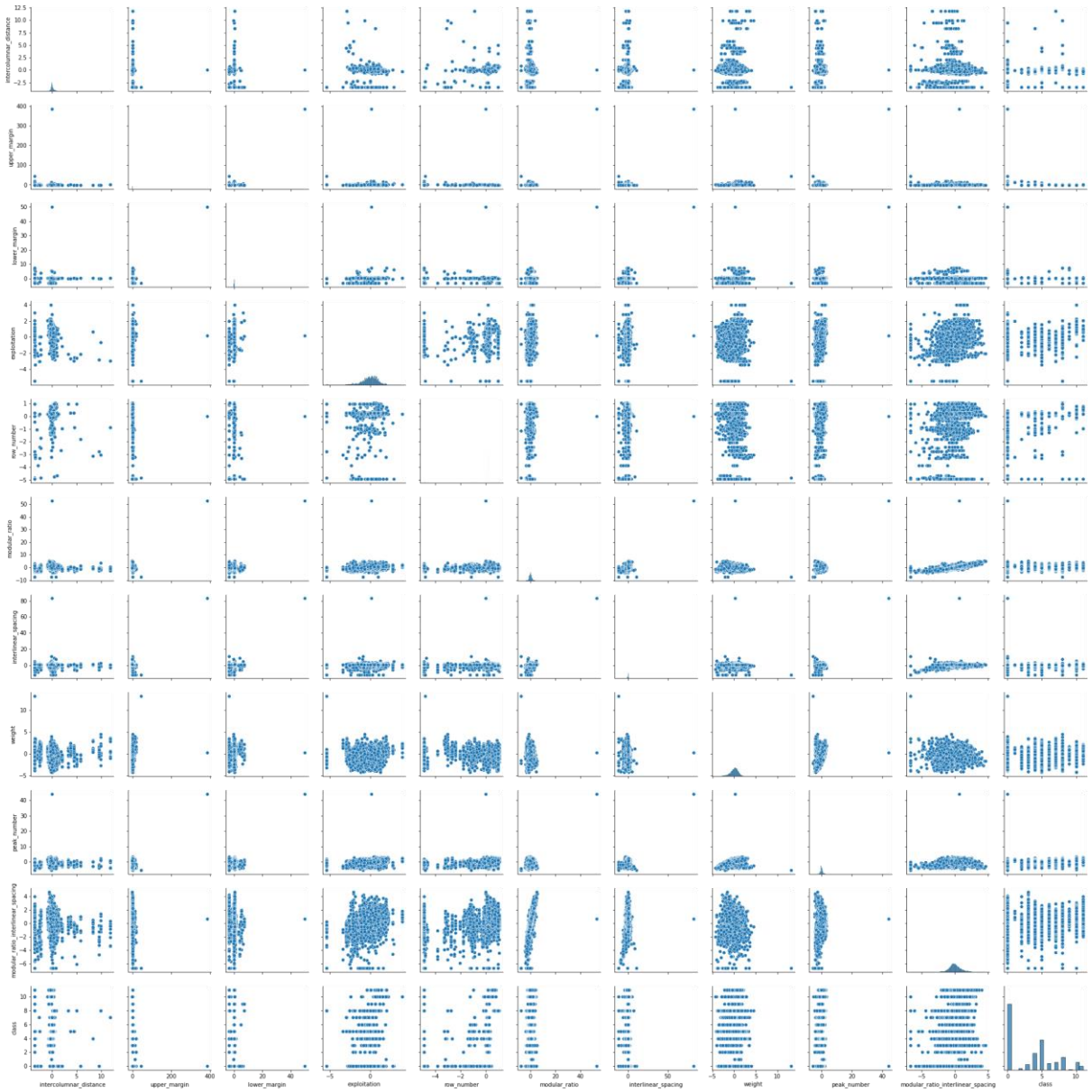

Figure 2.4: Correlation of Characteristics

The correlation map in Figure 2.5 reveals that peak number exhibits the strongest correlation with the target variable, followed by row number and modular ratio-intercolumnar spacing. Conversely, the target variable displays a strong negative correlation with modular ratio and interlinear spacing. Additionally, the table in Figure 2.5 highlights a nearly linear correlation between peak number and all the attributes, with the highest correlation being observed between the upper margin characteristics and interlinear spacing.

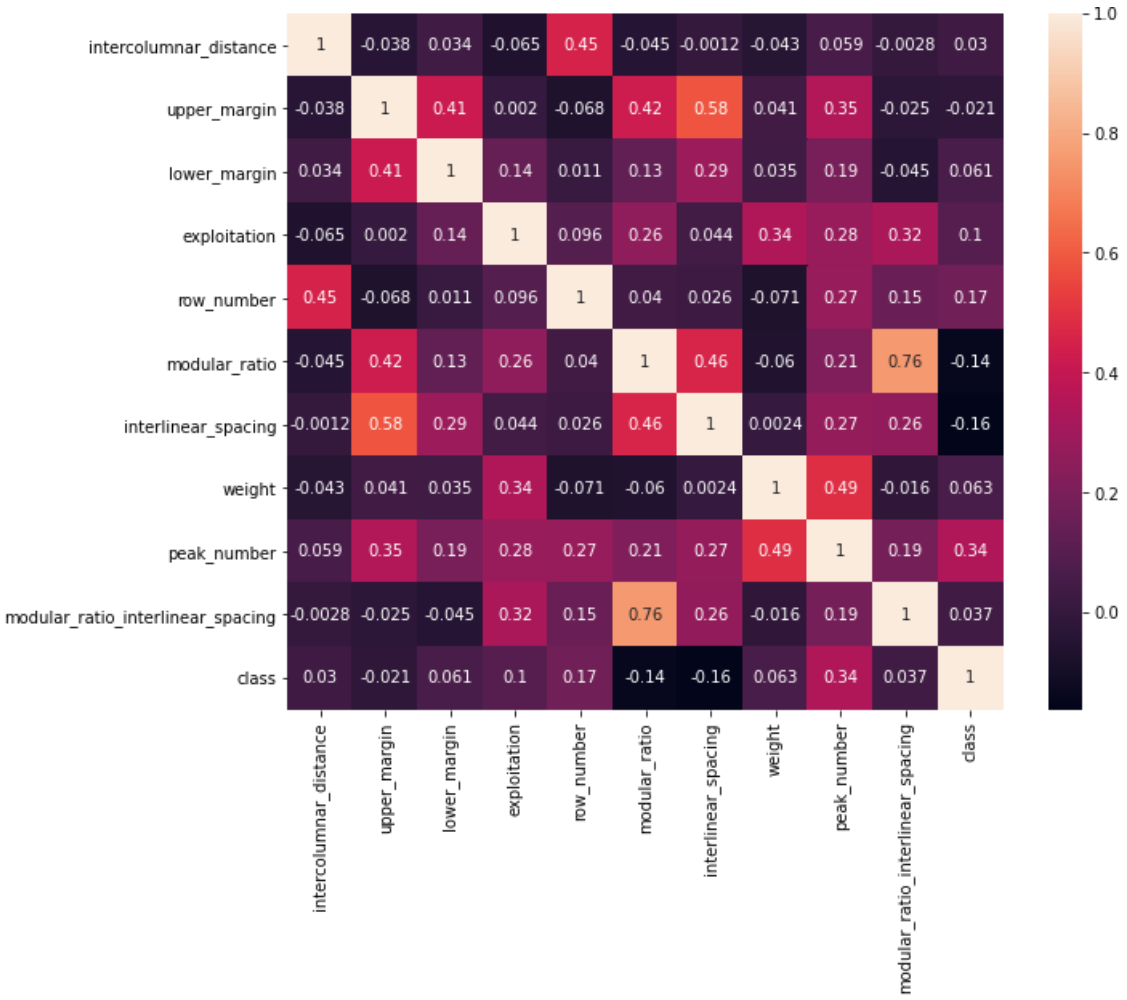

Figure 2.5: Correlation Map

Furthermore, by examining Figure 2.6, it is evident that outliers are present in each characteristic, as indicated by the boxplots. Specifically, the upper margin page feature displays a few measurements that are significantly distant from the other measurements. To eliminate these values, we will apply a data filtering approach based on this feature, retaining only samples whose standard deviation falls within the range of values [15, +15]. This method results in the exclusion of only one sample with a value that deviates significantly from the others. The updated boxplots are presented in Figure 2.7. Notably, all other features exhibit smooth measurements with no noteworthy issues or deviant values.

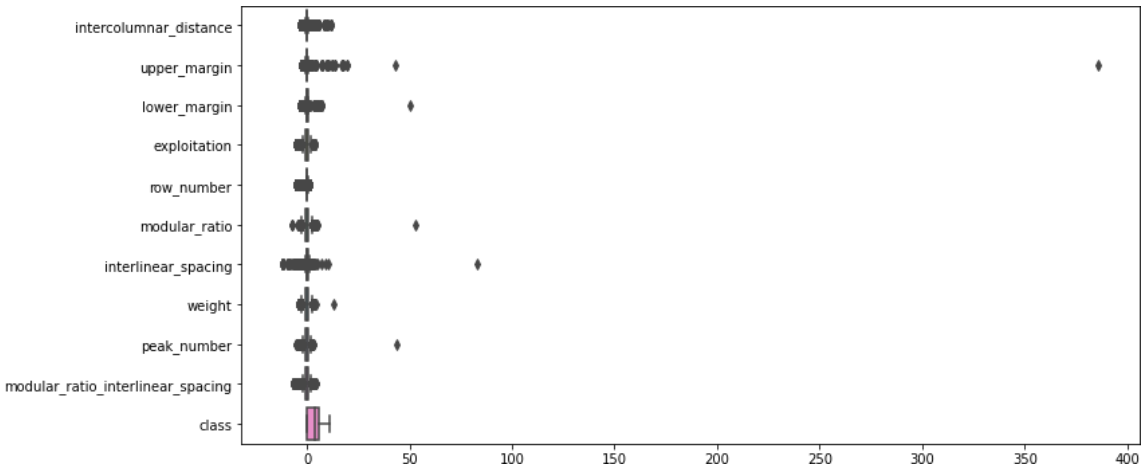

Figure 2.6: Boxplots before the removal of outliers

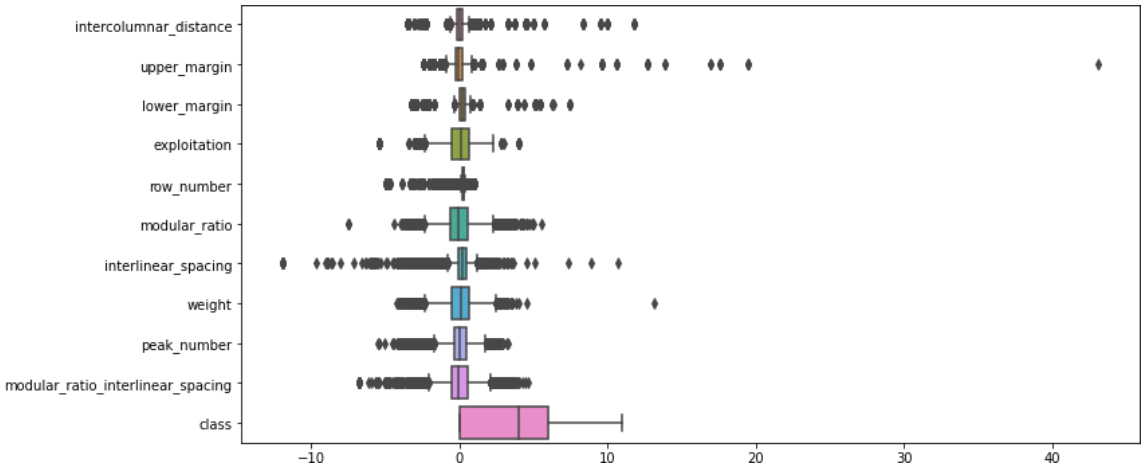

Figure 2.7: Boxplots after removal of outliers

Finally, we describe the pre-processing steps undertaken before utilizing the data to train and assess our models. Firstly, we segregated the data into feature (X) and target (Y) sets, with the former containing all characteristics except for the category, and the latter consisting solely of the category variable. Subsequently, we utilized the Standard Scaling normalization technique to normalize the resulting attribute data, ensuring that each attribute fell within the 0-1 value range by dividing by the standard deviation and subtracting the mean. To avoid potential bias from the test data, we fit and transformed the Standard Scaling technique solely on the training attributes, whereas only a transform was applied to the test attributes.

Additionally, we present a bar chart in Figure 2.8 depicting the distribution of the test set data in the output categories. As observed, the distribution is remarkably similar to that of the training set data shown in Figure 2.3, although the issue of uneven data distribution amongst categories remains. Nevertheless, the primary point of interest is that both the training and control sets possess comparable distributions.

During the model training and evaluation stages, we utilized all available features without implementing any feature selection or dimensionality reduction techniques, as we deemed that the available characteristics were relatively few in number and pertinent to our study.

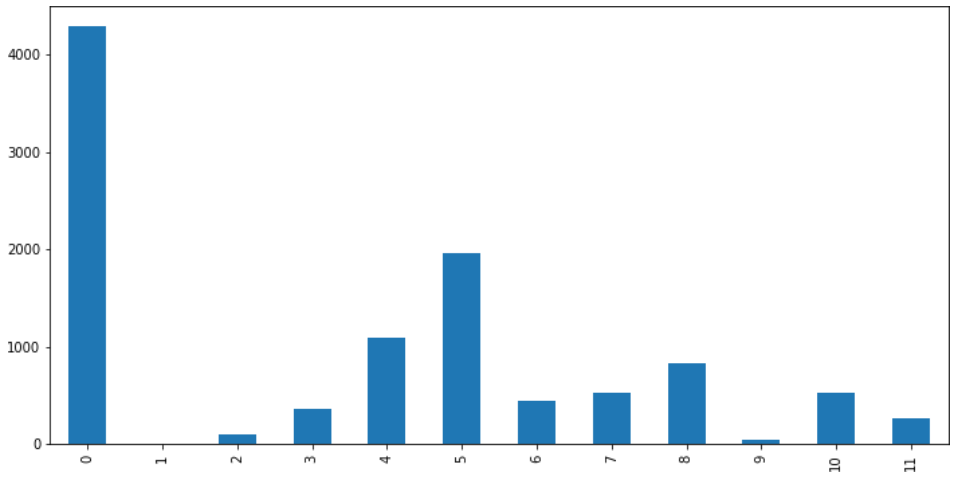

Figure 2.8: Distribution of categories (test set)

2.3 Development and Training of Machine Learning Models

Within this section, we will create, train, and evaluate various Machine Learning models that are widely recognized. As previously indicated, we will treat the task of forecasting the scribes of the Abilene Bible as a classification challenge. To that end, we will develop and train two distinct classification models.

2.3.1 Classification models

When we approach the problem as a classification problem, we assume that the variable we are trying to predict, the post-target variable, is a categorical variable that only takes on integer values from 0 to 11, representing the twelve categories of the Bible's twelve scribes. However, the distribution of the samples shows that there is an inequality in the twelve categories in both the training and test sets. This needs to be taken into account during the training process to create successful classifiers. To address the problem of data imbalance, we will use the balanced accuracy metric instead of the accuracy metric commonly used in these types of problems. The balanced accuracy metric is calculated as the average recall value for each category.

We will be using two models, a Naive Bayes Classifier and a Random Forest Classifier, both of which belong to the two categories of algorithms discussed in Chapter 1. In this case, we applied the grid search methodology to select the hyperparameters, but used the Balanced Accuracy metric instead of the regular Accuracy metric. The recall, which is the ratio of correct predictions to the sum of all predictions, represents the accuracy of the predictions for each category.

- **Naive Bayes Classifier:** For the Bayesian model, we utilized the grid search technique to determine the appropriate hyperparameter, var_smoothing, which is a normalization parameter added to the data variations to stabilize the solution. The resulting value for var_smoothing was 1e10, which was then utilized to train the classifier on the training data. The confusion map obtained by the Bayesian classifier is presented in Figure 2.9, while Table 2.3 provides a classification report for the different categories used in the classification.

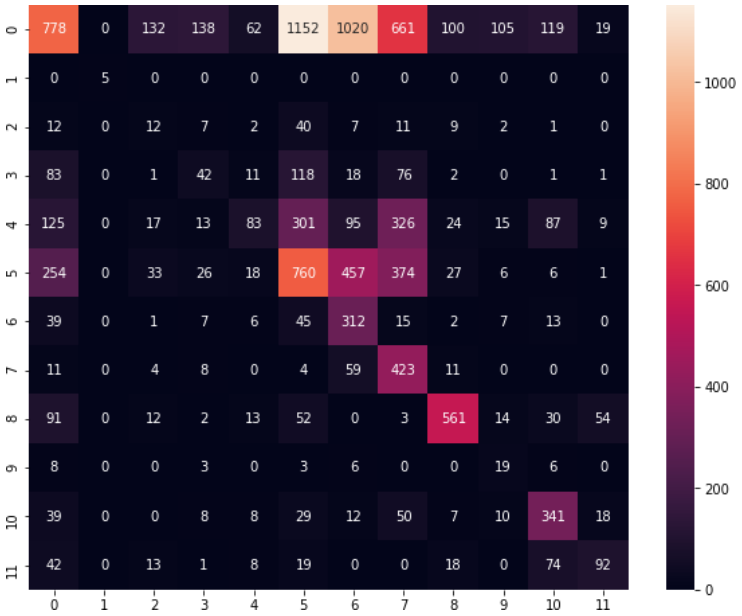

Figure 2.9: Random Forest Classifier Confusion Map

- **Random Forest Classifier:** To address the problem, we trained a Random Forest (RF) model. We used the grid search technique to identify the best hyperparameters, including the selection criterion for optimal splits, min_samples_split, and min_samples_leaf, as we did previously. Ultimately, we determined that the optimal values for criterion, min_samples_split, and min_samples_leaf were entropy, 3, and 1, respectively. We trained the model using these values on the training data and then calculated the Accuracy metric on the test data for the trained model.

Class	precision	recall	f1-score	support
0	0.52	0.18	0.27	4286
1	1.00	1.00	1.00	5
2	0.05	0.12	0.07	103
3	0.16	0.12	0.14	353
4	0.39	0.08	0.13	1095
5	0.30	0.39	0.34	1962
6	0.16	0.70	0.26	447
7	0.22	0.81	0.34	520
8	0.74	0.67	0.70	832
9	0.11	0.42	0.17	45
10	0.50	0.65	0.57	522
11	0.47	0.34	0.40	267
accuracy			0.33	10437
macro avg	0.39	0.46	0.37	10437
weighted avg	0.43	0.33	0.32	10437

Table 2.3: Classification Report of Naive Bayes Classifier

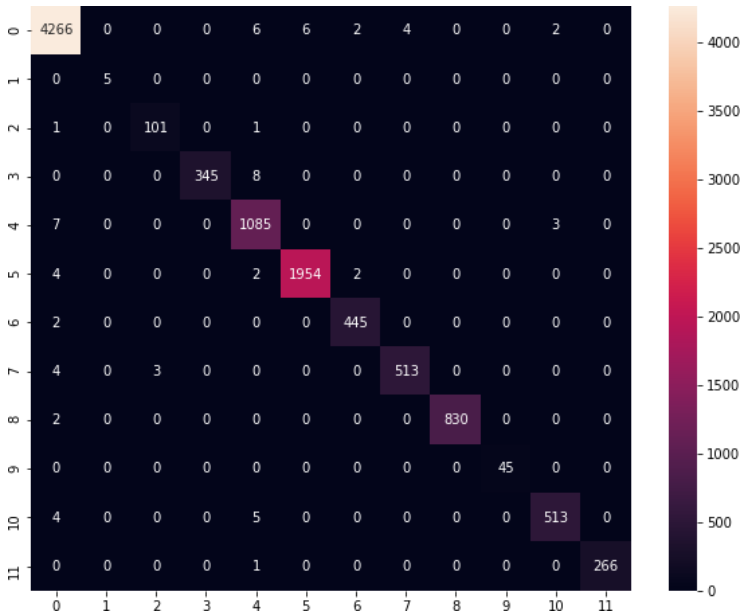

Figure 2.10: Naive Bayes Classifier Confusion Map

2.4 Comparison of results and conclusions

In this study, we evaluate the performance of two classification models that we trained. We present the Balanced Accuracy percentages of the models in a plot (Figure 2.11), where we can see that the Random Forest Classifier and Naive Bayes Classifier models achieved the desired values close to unity. We observe that the Random Forest Classifier model performs better than the Naive Bayes Classifier model in the classification problem. However, the Random Forest Classifier model takes longer to train and predict, as shown in Figure 2.12. Therefore, although the Random Forest Classifier model performs better, it comes at the cost of increased computational time.

Moreover, we compare our best-performing model with the optimal classifier proposed by the authors and dataset creators of the study [13]. Our Random Forest model achieves an Accuracy of 99.33% on the training set, while the optimal Decision Tree model proposed by De Stefano's group achieves an Accuracy of 98.25%. Thus, our best model outperforms the best model of De Stefano's group in this problem.

Class	precision	recall	f1-score	support
0	0.99	1.00	0.99	4286
1	1.00	1.00	1.00	5
2	0.97	0.98	0.98	103
3	1.00	0.98	0.99	353
4	0.98	0.99	0.99	1095
5	1.00	1.00	1.00	1962
6	0.99	1.00	0.99	447
7	0.99	0.99	0.99	520
8	1.00	1.00	1.00	832
9	1.00	1.00	1.00	45
10	0.99	0.98	0.99	522
11	1.00	1.00	1.00	267
accuracy			0.99	10437
macro avg	0.99	0.99	0.99	10437
weighted avg	0.99	0.99	0.99	10437

Table 2.4: Classification Report of Random Forest Classifier

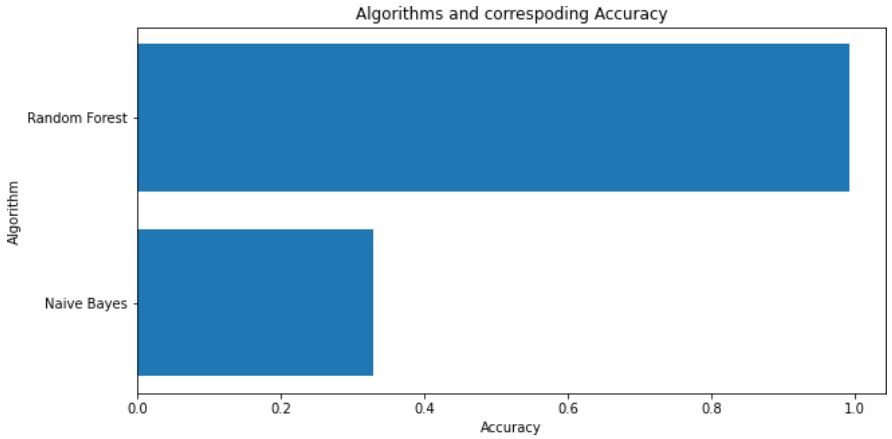

Figure 2.11: Algorithms and Corresponding Performance of the Accuracy Metric

In summary, our study demonstrates that the Random Forest decision tree algorithm generally outperforms conditional probability Bayesian models in the classification problem. However, considering the longer training and prediction time and the high computational resources required by the Random Forest models, they may not be the optimal choice. Thus, we emphasize that when comparing different models to choose the best one for a Machine Learning problem, it is crucial to consider not only the performance metrics of the prediction, but also other metrics such as time and hardware requirements (e.g., RAM, CPU, GPU) needed for training the models.

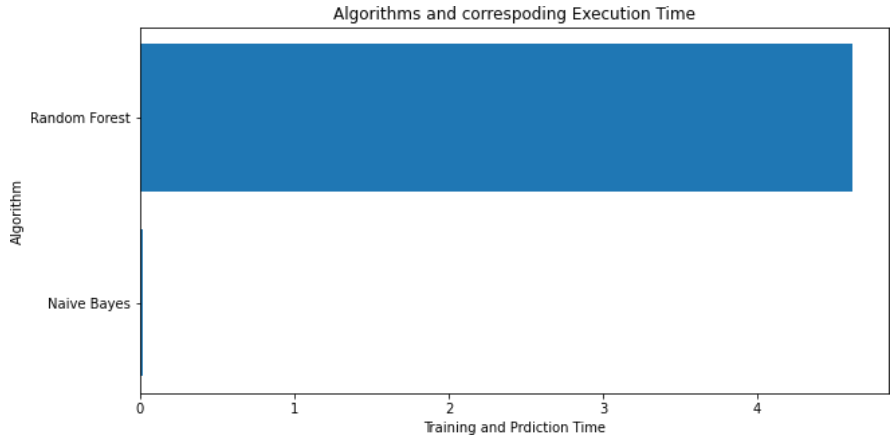

Figure 2.12: Algorithms and Corresponding Execution Times in Seconds

Appendix

Python code:

```
# -*- coding: utf-8 -*-
"""avila.ipynb

## Avila Bible
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import time

"""Download and extract data"""

! wget -O avilia.zip https:// archive.ics.uci.edu/ml/machine- learning-databases/00459/
    avila.zip
! unzip avilia.zip

names = ['intercolumnar_distance', 'upper_margin', 'lower_margin', '
    exploitation', 'row_number', 'modular_ratio',
                'interlinear_spacing', 'weight', 'peak_number', '
                    modular_ratio_interlinear_spacing', 'class']
df = pd. read_csv('avila/avila-tr.txt', names=names)
df.head()

"""Exploratory Data Analysis"""

# data description and information

print('Instances:', df.shape[0])
```

```
print(' Features:', df.shape[1])

print('\nInfo:')
df.info()

print('\nDescription:')
df. describe()

# check for null values

print('\nNULL:')
print(df.isnull().any())

# class distribution

df['class'] = pd. factorize(df['class'], sort=True)[0]
  # covert class labels from letters to ints
print(df['class']. value_counts(). sort_index())

plt.figure( figsize=(12 , 6))
df['class']. value_counts(). sort_index().plot(kind='bar')

# data distrinution

df.hist( figsize=(16 , 8), layout=(2 , 6), bins = 50)

# data feature correlation 1

sns. pairplot(df)

# data feature correlation 2

plt.figure( figsize=(10 , 8))
sns. heatmap(df.corr(), annot=True)

"""Prepr ocessing """#

outliers

plt.figure( figsize=(12 , 6))
sns. boxplot(data=df, orient='h')


df = df[np.abs(df. upper_margin - df. upper_margin.mean()) <= (15 * df.
    upper_margin.std())]
# keep only the ones that are within +15 to -15 standard
    deviations
```

```
plt.figure( figsize=(12 ,  6))
sns. boxplot(data=df,  orient='h')

df.info()

# split  train  data  to  features and labels
X_train  =  df.drop('class' ,  axis  =  1)
 y_train  =  df['class']

# import  test  data
df_test  =  pd. read_csv('avila/avila-ts.txt' ,  names=names)
df_test['class']  =  pd. factorize( df_test['class'],  sort=True)[0]
print( df_test['class']. value_counts(). sort_index())


plt.figure( figsize=(12 ,  6))
df_test['class']. value_counts(). sort_index().plot(kind='bar')

# splittest data to features and labels
X_test  =  df_test.drop('class' ,  axis  =  1)
y_test  =  df_test['class']

from  sklearn. preprocessing  import  StandardScaler

# data  scaling
scaler  =  StandardScaler()
scaler. fit_transform( X_train)  #  fit  scaler  only  on  traing
scaler. transform(X_test)

"""Classification"""

from  sklearn. model_selection  import  GridSearchCV
from  sklearn. metrics  import  classification_report , confusion_matrix

accs =[]
accs_without_opt  =[] names2
=[]
times2  =[]

from  sklearn. naive_bayes  import  GaussianNB
params  =  {'var_smoothing' :  np. linspace(1e-10,  1e-8,  5)}
gnb  =  GaussianNB()
clf  =  GridSearchCV(gnb,  params ,  scoring=('balanced_accuracy'))
clf.fit(X_train ,  y_train)
```

```
print(clf. best_params_)

start_time = time.time()
best_clf = clf. best_estimator_
best_clf.fit(X_train , y_train)

acc = best_clf.score(X_test , y_test) accs.append(acc)
names2 .append(' Naive Bayes' )

t = time.time() - start_time times2
.append(t)
print("fit predict: %s seconds" % (t))
    print(' Accuracy' , acc)

clf_s = gnb clf_s.fit(X_train,y_train)
acc = clf_s.score(X_test , y_test) accs_without_opt.append(acc)

print(' Without Optimization Accuracy' , acc) y_pred =

best_clf. predict(X_test)

print(' Classification Report: \n' , classification_report(y_test,y_pred))
print(' Confusion Matrix: \n' )
plt.figure( figsize=(10 , 8))
sns. heatmap( confusion_matrix(y_test , y_pred), annot=True, fmt=' d' )

train_acc = best_clf.score(X_train , y_train)
print(' Training Score: ' , train_acc)
test_acc = best_clf.score(X_test , y_test)
print(' Testing Score: ' , test_acc)

from sklearn. ensemble import RandomForestClassifier
import warnings
warnings. filterwarnings(' ignore' )

params = {' criterion' : [' gini' , ' entropy' ], '
                min_samples_split' : np.arange(1 ,5), '
                min_samples_leaf' : np.arange(1 ,5)
}

rfc = RandomForestClassifier()
```

```
clf = GridSearchCV(rfc, params , scoring=(' balanced_accuracy' )
    )
clf.fit(X_train , y_train)

print(clf. best_params_)

start_time = time.time()
best_clf = clf. best_estimator_
best_clf.fit(X_train , y_train)

acc = best_clf.score(X_test , y_test) accs.append(acc)
names2 .append(' Random Forest' )

t = time.time() - start_time times2
.append(t)
print("fit predict: %s seconds" % (t))
   print(' Accuracy' , acc)

clf_s = rfc clf_s.fit(X_train , y_train)
acc = clf_s.score(X_test , y_test) accs_without_opt.append(acc)

print(' Without Optimization Accuracy' , acc) y_pred =

best_clf. predict(X_test)

print(' Classification Report: \n' , classification_report( y_test , y_pred))
print(' Confusion Matrix: \n' )
plt.figure( figsize=(10 , 8))
sns. heatmap( confusion_matrix(y_test , y_pred), annot=True, fmt=' d' )

train_acc = best_clf.score(X_train , y_train)
print(' Training Score: ' , train_acc)
test_acc = best_clf.score(X_test , y_test)
print(' Testing Score: ' , test_acc)

plt.figure( figsize=(10 ,5))

plt.barh(names2 , accs)
plt.ylabel(' Algorithm' )
plt.xlabel(' Accuracy' )
plt.title(' Algorithms and correspoding Accuracy' )
```

```
plt.figure( figsize=(10 ,5))

plt.barh(names2 ,  times2 )
plt.ylabel(' Algorithm' )
plt.xlabel(' Training  and  Prdiction  Time' )
plt.title(' Algorithms  and  corresponding  Execution  Time' )
```

Bibliography

[0] W. A. Belson, "Matching and Prediction on the Principle of Biological Classification," *Journal of the Royal Statistical Society Series C*, vol. 8, no. 2, pp. 65–75, June 1959. [Online]. Available: https://ideas.repec.org/a/bla/ jorssc/v8y1959i2p65-75.html

[1] Decision tree learning, "Decision tree learning — Wikipedia, the free encyclopedia," 2011, [Online; accessed 15-May-2021]. [Online]. Available: https://en.wikipedia.org/wiki/Decision_tree_learning

[2] J. R. Quinlan, "Induction of decision trees," *Mach. Learn.*, vol. 1, no. 1, p. 81–106, Mar. 1986. [Online]. Available: https://doi.org/10.1023/A: 1022643204877

[3] ——, *C4.5: Programs for Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1993.

[4] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Confer- ence on Document Analysis and Recognition*, vol. 1, 1995, pp. 278–282 vol.1.

[5] L. Breiman, "Random forests," *Mach. Learn.*, vol. 45, no. 1, p. 5–32, Oct. 2001. [Online]. Available: https://doi.org/10.1023/A:1010933404324

[6] S. M. Piryonesi, *The Application of Data Analytics to Asset Management: Deteri- oration and Climate Change Adaptation in Ontario Roads*, 2019.

[7] H. L. SEAL, "Studies in the History of Probability and Statistics. XV The historical development of the Gauss linear model," *Biometrika*, vol. 54, no. 1-2, pp. 1–24, 06 1967. [Online]. Available: https: //doi.org/10.1093/biomet/54.1-2.1

[8] J. Cramer, "The Origins of Logistic Regression," Tinbergen Institute, Tinbergen Institute Discussion Papers 02-119/4, Dec. 2002. [Online]. Available: https://ideas.repec.org/p/tin/wpaper/20020119.html

[9] F. Rosenblatt, *The Perceptron, a Perceiving and Recognizing Automaton Project Para*, ser. Report: Cornell Aeronautical Laboratory. Cornell Aeronautical Laboratory, 1957. [Online]. Available: https://books.google.gr/books?id=P_ XGPgAACAAJ

[10] W. Mcculloch and W. Pitts, "A logical calculus of ideas immanent in nervous activity," *Bulletin of Mathematical Biophysics*, vol. 5, pp. 127–147, 1943.

[11] M. Minsky, "Steps toward artificial intelligence," *Proceedings of the IRE*, vol. 49, no. 1, pp. 8–30, 1961.

[12] C. De Stefano, M. Maniaci, F. Fontanella, and A. Scotto di Freca, "Reliable writer identification in medieval manuscripts through page layout features: The "avila" bible case," *Engineering Applications of Artificial Intelligence*, vol. 72, pp. 99–110, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0952197618300721

[13] K. K. Sharma and A. Seal, "Spectral embedded generalized mean based k-nearest neighbors clustering with s-distance," *Expert Systems with Applications*, vol. 169, p. 114326, 2021. [Online]. Available: https: //www.sciencedirect.com/science/article/pii/S0957417420310186

[14] I. Baidari and N. Honnikoll, "Accuracy weighted diversity-based online boosting," *Expert Systems with Applications*, vol. 160, p. 113723, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/ S0957417420305479

[15] J. Robledano Arillo, "Quality control of digital representations of manuscript texts: proposal of a standards-based method," *Open Information Science*, vol. 5, pp. 27–44, 06 2021.