Department of Computer Engineering

Bu-Ali Sina University

Machine Learning Course

Machine Learning 1st Assignment

By:

Alireza Maleki

Course Professor:

Professor Muharram Mansoorizadeh

Spring 2023-24

TABLE OF CONTENTS

1 INTRODUCTION

The iris dataset is one of the classic data sets used for testing machine learning classification algorithms. Originally published by Ronald Fisher in 1936, it contains 4 measurements for 150 iris flowers consisting of 50 samples from each of 3 related iris species—Iris setosa, Iris versicolor, and Iris virginica. The challenge is to use the petal and sepal width and length measurements to correctly predict the iris species with an unseen test set. This makes it an ideal simple dataset to get experience implementing and evaluating major classification models including Naïve Bayes, Decision Trees, Random Forest, and Neural Networks.

In this project, we tackle the iris species classification problem using scikit-learn, Python's main machine learning library. All models are implemented programmatically and evaluated for their generalization accuracy using the sklearn model workflow of model declaration, training, predictions, and scoring. In addition to comparing out-of-the-box model accuracy, we explore the effects of key hyperparameters like tree depth and number of estimators for each algorithm class. The full experimental code is developed interactively using Google Colaboratory to enable rapid building, testing, and analysis.

The goals of this project are two-fold: from an applied machine learning perspective, it allows benchmarking of the major classifiers on a fundamental signal classification task. From an educational viewpoint, it provides exposure to training, evaluating, comparing, and tuning machine learning models programmatically using the versatile scikit-learn API. The end result is a solid hands-on foundation in both machine learning concepts and coding skills using real-world curated datasets as the basis.

2 METHODOLOGY

2.1 Dataset

The iris dataset is a classic for testing classification algorithms with only 150 instances split evenly across its 3 classes. The small size makes it easy to overfit with high model capacity. The measurements are continuous, not requiring any encoding. Overall, it served as a good basic benchmark for comparing the major classification models but was too small to take advantage of neural networks. In a nutshell, This dataset includes 150 records. each instance has 4 attributes (sepal length, sepal width, petal length, and petal width) to classify between 3 classes (Iris-setosa, Iris-virginica, and Iris-versicolor).

2.2 Data Pre-processing

2.2.1 Splitting Data

Since data pre-processing plays an important role in machine learning, we are supposed to do some in our dataset. our main preprocessing is divided into splitting data into two parts: training data and testing data. In machine learning, training data constitutes the portion of a dataset employed to teach a model by adjusting its parameters based on input-output pairs. The primary objective is for the model to discern patterns and relationships within the data. In contrast, test data, a distinct subset untouched during training, serves the crucial role of evaluating the model's generalization performance on new, unseen examples. The division into training and test data is essential for gauging whether the model has learned patterns or is merely memorizing the training set, helping to ensure its effectiveness in real-world scenarios. The randomization of data samples and the use of performance metrics further contribute to a robust assessment of the model's capabilities.

In this assignment, we split the data into a 20% testing dataset and an 80% training dataset. this operation was done easily using the scikit-learn function, $train\_test\_split$. As the total instances are 150 in this dataset, according to the division ratio, training data and testing data consist of 120 and 30 samples.

2.2.2 Categorical Encoding

Categorical encoding transforms textual categorical features into numeric representations so machine learning models can interpret categories they normally couldn't process directly. Choosing the right encoding that matches the data characteristics is crucial so models can properly leverage categoricals. The goal of encoding is to enable models to understand categories by converting text to informative numbers while balancing semantic meaning, complexity, dimensionality, and information loss through the representation. Since our dataset classes are in categorical format, we need to encode them into numerical to be understandable for machines. So that, the three main classes are encoded to Iris-setosa (0), Iris-virginica (1), and Iris-versicolor (2).

2.2.3 Feature Scaling

Feature scaling is an important preprocessing step in machine learning that rescales features to a common range of values, often between 0 and 1. It is used to standardize the levels of features before training a model. Without feature scaling, variables with wider ranges of values can unduly influence model calculations compared to variables with narrower ranges.

Table 1: Random Forest Hyperparameters tuning based accuracy.

Number of Trees	1	3	6	9	10
Accuracy	90.0%	93.33%	96.67%	96.67%	100.0%

Scaling removes misleading effects from variation in units and ranges in the raw data. It gives equal importance to all features, prevents weighting issues from imbalanced variables, and improves model convergence and stability. Overall feature scaling enables machine learning algorithms to accurately model the data by avoiding unintended dominance from arbitrary differences in input variables. Transforming features to common scales allows the discovery of true patterns.

2.3 Random Forest Classifier

Basically, Random Forest is an algorithm based on decision trees. As obvious from the name, this algorithm consists of some decision trees together. For creating this classifier, the first step is to consider the dataset and build a bootstrap dataset from the original one. The bootstrap dataset should be the size of the original data. Also, the new temporary dataset is allowed to have Repetitious records. Every decision tree that is a member of our forest based on this bootstrap dataset which is a second step. Note that each decision tree will grow according to some features of the original dataset, not all the features. The third step is to create multiple decision trees and then consider their classification as a vote. For the final step, after voting from each tree, the results should be set as the most votes. A random forest model was built using sci-kit-learn's RandomForestClassifier. The number of trees in the forest was set to 10. All other parameters were left as default. The model was trained on 80% of the data and tested on the remaining 20%. With the default parameters, it achieved an accuracy of around 100% on the test set on at least 10 estimators or in other words 9 decision trees. Tuning the max depth parameter, which controls how deep each decision tree can grow, did not lead to noticeable improvements in accuracy. The random forest is quite robust to overfitting with many estimators. So based on this knowledge, our main focus for hyperparameter tuning should be on the number of estimators that completely affect the accuracy we are supposed to achieve. the table below completely demonstrates our accuracy score based on the number of trees.

2.4 Decision Tree Classifier

Decision Tree classifier is one of the supervised learning algorithms for classification. To classify, this algorithm creates a tree for decisions in various conditions. The tree has some specific characteristics explained in the following: the internal nodes or branches represent a feature(or attribute), the branch represents a decision rule, and each leaf node represents the outcome.

The challenging task we faced in decision trees is which features should be chosen between the existing features. To choose that, we are going to detect which features classify the final class more accurately. After classifying, we need to detect which leaf is impure and which leaf classifies 100% correctly. One way is to choose a feature with the least impure leaves while the other way is to use Gini impurity or information gain. For categorical features we need to calculate the Gini for each leaf and compare only if the number of entities is equal, otherwise, we head to calculate the total Gini impurity which is a weighted average for Gini impurities for the leaves. After calculating the Gini for every feature, the feature with the least Gini value is chosen to be on the root for example.

A decision tree classifier was trained using Scikit-learn's DecisionTreeClassifier. The model was trained with no initialization for the depth of the tree because the nodes are expanded until all leaves are pure. The least possible value for this hyperparameter is 3 which provides the 100% accuracy. The max depth parameter specifies the maximum depth or level that a decision tree can grow to before stopping. Max depth directly impacts model complexity and performance. Optimal max depth leads to balanced accuracy vs overfitting while larger depths improve accuracy but reduce speed and interoperability. the model criterion was set to $Gini$ as explained above. With these parameters, the accuracy on the test set was 100%. The decision tree and random forest both achieved good accuracy but the decision tree was more prone to overfitting compared to the random forest ensemble method.

2.5 Naive Bayes Classifier

Naive Bayes is a simple but surprisingly effective supervised machine learning algorithm used primarily for classification problems. It is based on Bayes' theorem, which calculates conditional probability, but makes the "naive" assumption that all input features are independent. A Naive Bayes classifier first learns the conditional probability of each input feature belonging to each target class from the training data. Then, when applied to new unlabeled data, it multiples these learned conditional probabilities for the input features and selects the class with the highest combined conditional probability as the predicted output. Despite its simplicity and the obviously inaccurate assumption of feature independence, Naive Bayes often competes well with more advanced classifiers due to its efficient use of small or sparse training data. It performs particularly well for natural language processing tasks like spam filtering, sentiment analysis and document classification.
A Gaussian naive Bayes model was used from sci-kit-learn, with default parameters. Being a simpler probabilistic model, its accuracy was lower compared to the other models, achieving 96.67% on the test set. Note that, Naive Bayes accuracy cannot be improved by parameter tuning, given its

simple statistical assumptions. But it served as a baseline benchmark.

2.6 Neural Networks

Neural networks are computing systems inspired by the biological neural networks that constitute animal brains. Neural networks are composed of nodes, which loosely model neurons, connected to form a network. Data comes into the input nodes, flows through the network transforming as it travels along weighted connections, and outputs a prediction or classification from the output nodes. Neural networks can be trained to perform complex pattern recognition and machine learning tasks without being explicitly programmed for the task.

For an iris dataset that contains measurements of sepals and petals for three iris species, a simple feedforward neural network could be constructed with 4 input nodes representing the 4 features, 3 output nodes representing the 3 species, and one or more hidden layers in between. This allows the network to learn nonlinear combinations of the measurements that best separate and classify the different iris types. The network could have a single hidden layer or multiple hidden layers, with varying numbers of nodes in each, depending on the desired model complexity. According to our experiments, the most accurate structure for the neural network is to have 3 hidden layers consisting of at least 7 neurons in each layer.

Determining the optimal number of hidden layers and nodes is largely empirical and depends on the specific dataset and classification task. Networks with more layers and nodes have higher complexity and can model more intricate patterns in the data, but may be more prone to overfitting. Common practice is to train several networks with differing numbers of layers and node counts and compare their accuracy on a test set, choosing the one with the best testing performance. The number of training iterations, controlled by max_iter, also affects overfitting - more iterations allow better convergence but can overfit, while early stopping provides regularization which approximately equals 350 iterations. This number was chosen to avoid both overfitting and achieving the best accuracy based on the hidden layers. Tuning these hyperparameters is key to achieving high accuracy while avoiding overfitting.

A simple neural network was built using Scikit-learn's MLPClassifier model. The model had three hidden layers with 7 nodes in each layer and used the $'ReLU'$ activation function as default for weight optimization. The test accuracy was 100%, similar to the decision tree classifier. Increasing the number of hidden nodes or layers did not noticeably improve performance, likely due to the small number of training examples. Scikit-learn provides a convenient Neural Network model

for basic use cases. With the limited data, it achieved comparable performance to other non-ensemble models. However, unlike Keras/TensorFlow, scikit-learn does not support more complex neural network architectures.

3 CONCLUSION

In conclusion, the main goal of this assignment is to implement some supervised algorithms and understand the concept of each. For testing our models iris dataset is suggested. the dataset was divided into train and test sets for training and evaluating the models. Some preprocessing tasks such as data splitting, feature scaling, and categorical encoding were done to get the data completely ready for training. Of the models tested, Random Forest, neural networks, and decision tree performed the best, followed by Naive Bayes. The ensemble method of Random Forests prevents overfitting, while Naive Bayes has inherent regularization being a simpler probabilistic model. Decision trees and neural networks had similar accuracy but higher variance. Tuning model hyperparameters had little effect owing to the small size of this well-known dataset. All models achieved over 96.63% accuracy, with random forest, neural networks, and decision tree highest at 100%.