Data Classification using the Random Forest Method

1st Lukáš Patrnčiak
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
xpatrnciak@stuba.sk

2nd Andrej Tomčík
Faculty of Electrical Engineering and Information Technology
Slovak University of Technology in Bratislava
Bratislava, Slovakia
xtomcik@stuba.sk

***Abstract*—This paper focuses on the problem of data classification using decision trees and Random Forests. The theoretical background of decision trees, entropy and information gain as the basic principles of classification is described. Subsequently, a custom decision tree-based classifier is implemented as well as an ensemble Random Forest model that improves robustness and accuracy through bagging. The task of the models was to predict the ticket price category based on various attributes. The results are evaluated using accuracy and confusion matrices, with Random Forest demonstrating higher classification accuracy compared to the standalone decision tree.**

***Index Terms*—decision tree, random forest, machine learning**

I. Introduction

Decision trees and ensemble methods, such as Random Forest, are among the most commonly used classification algorithms in machine learning. Their popularity lies in their ease of decision making interpretability, ability to work with heterogeneous data, and high accuracy in practice.

This paper focuses on the task of classifying ticket price categories based on input data such as ticket type, number of transfers, time to departure, and other relevant attributes. The main objective was to develop and implement custom versions of the decision tree and Random Forest method without using predefined library models.

The paper provides a detailed description of decision tree theory, entropy and information gain computation, as well as the bagging method that forms the basis of the Random Forest algorithm.

After processing the dataset, the models were trained on the categorical data and their performance was evaluated using accuracy and confusion matrix analysis. The results show that the ensemble Random Forest approach achieves higher accuracy compared to a single decision tree, confirming its advantages in classification tasks with heterogeneous data.

II. Theoretical Background

Decision trees as well as random forests can be formally represented using directed graphs. A graph $G = (V, E)$ consists of a set of vertices V and a set of edges E. A tree is a special type of continuous, acyclic graph in which there is exactly one path between every two vertices. A root tree is a tree in which one vertex is labeled as the root and the other vertices form a hierarchical structure of leaves and nodes [1].

Such a structure is suitable for representing, e.g., algebraic expressions or decision processes, where each internal node represents a decision condition and the leaves correspond to the outputs of the model [2].

A. Decision Tree

A decision tree is a method for approximating discrete objective functions, where the learned model takes the form of a tree structure consisting of questions (attribute constraints) and answers (branches) that lead to a decision (a leaf). The tree can be reformulated in the form of "if-then" rules to increase its interpretability.

Decision tree properties:

- They work with data in the form of attribute-value pairs.
- They support multi-class classification.
- They are robust to noise and missing values.
- They allow disjunctive (or) class descriptions.

Entropy: $H(S)$ denotes entropy and it indicates expresses the uncertainty or impurity of a set of examples. In the context of classification, it determines the degree of "mixing" of classes in a given set:

$$H(S) = -\sum_{i=1}^{n} p_i \log_2 p_i, \tag{1}$$

where p_i is the probability of occurrence of examples of class i in the set S.

Information gain: Indicates how much the entropy of the set is reduced after partitioning according to the attribute A:

$$\text{Gain}(S, A) = H(S) - \sum_{v \in Values(A)} \frac{|S_v|}{|S|} \cdot \text{Entropy}(S_v). \quad (2)$$

The attribute that maximizes the entropy reduction is selected - thus achieving the cleanest distribution.

Decision Tree Algorithm: Decision trees are typically constructed using the ID 3 (Iterative Dichotomiser 3) algorithm, which:

- selects the best attribute based on information gain,
- recursively builds a tree by branching based on attribute values,
- terminates branching if the data is homogeneous or the attributes are exhausted.

B. Random Forest

Random Forest is an ensemble method that creates multiple decision trees and combines their outputs. Each tree is trained on a different bootstrap sample (resampling with repetition), thus reducing the variance of the model. The resulting prediction is determined by majority voting [3].

Random Forest algorithm:

1) Bootstrap samples are randomly created from the training data.
2) Each tree is trained on a different sample and uses only a random subset of attributes.
3) For a new input, the predictions of all trees are aggregated by voting.

This approach reduces re-learning, increases robustness to noise, and allows efficient handling of large datasets.

Example. If 5 trees predict the classes Low, Medium, Low, High, Low, the resulting prediction is Low based on the majority vote.

III. Data Processing and Analysis

A. Dataset Description

For the experiments, a dataset in .csv format was used, containing information about commercial airline tickets. The original dataset contained 50,000 records, of which 36,638 samples remained after cleaning and removing incomplete or duplicate data. Each record contained the following attributes:

- Airline,
- Flight - flight identifier,
- Source city, Destination city - departure and arrival city,
- Departure time, Arrival time - departure and arrival time frames,
- Stops - number of transfers,
- Class - travel class (Economy/Business),
- Duration - flight length,
- Days left - number of days left until departure,
- Price - ticket price (target variable).

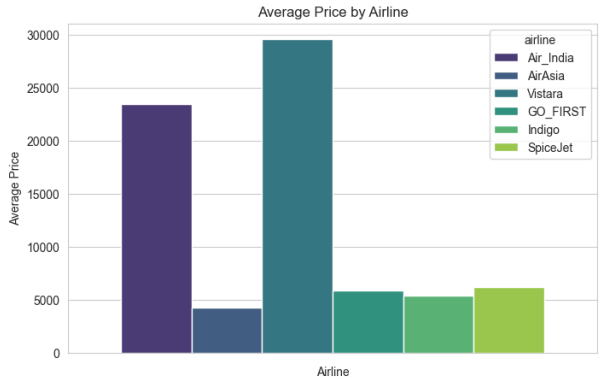

Fig. 1. Graph showing the relationship between average price and airline

- Columns ID and Flight were marked as irrelevant for classification and subsequently removed.

B. Data Cleaning and Preparation

At the beginning of the process, the data was analysed for:

- missing values,
- duplicate records,
- anomalous numerical values.

This preliminary analysis allowed to identify unnecessary or problematic data and subsequently clean the dataset. The resulting dataset thus consisted exclusively of valid, relevant records suitable for training classification models.

C. Numeric Attribute Categorization

Since the decision tree used requires discrete inputs, the numerical attributes were categorized based on a quantile distribution into three groups:

- Price - low, medium, high
- Duration - short, medium, long
- Days left - last, soon, flexible

In this way, the compatibility of the data with the implemented decision algorithm was ensured without significant loss of information value.

D. Data Visualization

As part of the Exploratory Data Analysis (EDA), a visualisation of the average price by airline was created (Figure 1). This visualization provided an overview of the distribution of the target variable and its relationship to each category of input attributes.

E. Data Allocation

The dataset was then divided into:

- input set X - all attributes except price,
- target set Y - the price value.

For the purpose of model evaluation, the data was split into:

- training set (70%) and
- test set (30%).

A fixed value of the parameter `random_state` was used in the partitioning to ensure reproducibility of the results across multiple runs of the model.

F. Tools Used

The following Python programming language libraries were used for implementation and data processing.

- NumPy - numerical calculations,
- Pandas - manipulation of tabular data,
- Seaborn, Matplotlib - visualizations and graphical outputs,
- Scikit-learn - data partitioning and plotting confusion matrices,
- collections, dataclasses - working with structured objects.

IV. Algorithm Application

This section describes the design, implementation and functionality of the classification models created: decision tree and Random Forest method. Both models were designed and implemented without using off-the-shelf library classifiers, with an emphasis on understanding the internal logic of learning and prediction.

A. Decision Tree

The implementation of the decision tree was based on two main classes: the TreeNode, representing the tree node, and the DecisionTree, which provides the learning, branching and prediction process.

Structure of node (`TreeNode`):

- attribute: the attribute by which the node is branched (for internal nodes only),
- children: a map of attribute values to other nodes,
- value: predicated class (only for leaf nodes).

Calculation of entropy and information gain: The branching of the tree is controlled by the entropy and information gain calculation:

- Entropy expresses the degree of impurity of the classes in the dataset,
- Information gain determines which attribute best reduces the entropy after partitioning.

The entropy calculation is carried out using the equation 1 and in the case of information gain, it is the equation 2.

Architecture of the Model: `DecisionTree` class contains:

- `__init__`: initializes the maximum tree depth, random number generator and the ability to limit the number of available attributes (`max_features`),
- `fit()`: trains the tree on data X and target values Y,
- `_build_tree()`: recursively builds the tree structure - terminates when outputs are homogeneous, attributes are exhausted, or maximum depth is reached,
- `_best_attribute()`: selects the attribute with the largest information gain (optionally from a random subset),
- `_predict_row()` and `predict()`: predict the class for one or more input samples.

For unknown attribute values during prediction, fallback is used: return to the most frequent class in the training data.

Visualization: The model also includes text visualization using `visualize()`, which recursively displays the tree in hierarchical form. It shows both the decision nodes and the predicted values in the leaves.

Output: The model predictions were exported to a CSV file containing pairs: actual value - predicted value. This output was then used for comparison with the Random Forest results.

B. Random Forest

The Random Forest model is an ensemble classifier that combines multiple independently trained decision trees. Each tree:

- trains on a bootstrap sample of the original dataset (with backtracking),
- only works with a random subset of attributes, increasing diversity.

Architecture of the Model: `RandomForest` class implements:

- `__init__`: setting the number of trees (`n_estimators`), maximum depth (`max_depth`), random seed (`random_state`) and initialization of internal structures (tree lists, RNG generator),
- `fit()`: creates `n_estimators` of separate trees:
  - each on its own bootstrap sample of training data,
  - each can only use a limited number of attributes (if `max_features` is set),
  - individual trees are trained using the already defined `DecisionTree` class.

Prediction: `predict()` method:

- sends the entire test dataset to each tree,
- collects predictions from all trees for each input,
- applies majority voting to each input and determines the final class.

$$\hat{C}_{\text{rf}}^{B}(x) = \text{majority vote} \left\{ \hat{C}_b(x) \right\}_{b=1}^{B}$$

Output: Similar to the decision tree, the prediction results were exported to a CSV file containing the actual and predicted values.

V. Evaluation of Results

A. Accuracy Metrics

The performance of the models was evaluated using the accuracy and error metrics, which expresses the proportion of correctly predicted cases to all samples:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{All predictions}},$$

$$\text{Accuracy} = 1 - \text{Accuracy}$$

	Accuracy	Error
Decision Tree	86.54%	13.46%
Random Forest	87.81%	12.19%

TABLE I
COMPARISON OF ACCURACY AND ERRORS OF DECISION TREE AND RANDOM FOREST

Accuracy was calculated using the internal method_`accuracy_score(y_true, y_pred)`, which compares the predicted classes with the real values. This method has been built into the broader evaluation method `score()`, which returns either an accuracy or an error rate based on the parameter `return_error`.

B. Summary

Both models were implemented in a modular way, with emphasis on readability, interpretability and parameter tunability. Decision Tree provides an interpretable and fast model, while Random Forest increases robustness and accuracy by aggregating the predictions of multiple weaker models.

C. Results of the Models

The models reached the following values: Random Forest outperformed the decision tree by approximately 1.27 %, confirming the benefit of the ensemble approach and bagging in increasing the robustness and generalization ability of the model.

D. Confusion Matrix

A more accurate analysis of the results was performed using confusion matrices to visualize the number of correct and incorrect predictions for each class [4].

- The lines represent the actual classes (actual),
- Columns represent predicted classes (predicted),
- The main diagonal contains the number of correctly classified cases.

Class designations (ticket prices):

- 0 – low,
- 1 – medium,
- 2 – high.

For example, the decision tree misclassified 698 cases of class high as medium. In contrast, Random Forest showed improvement in classification accuracy for all classes - especially for the medium and high classes, where errors between adjacent classes were reduced.

E. Summary

Confusion matrices and numerical metrics confirm that Random Forest provides a more robust and accurate model, especially when the target variable contains multiple discrete classes. Thus, the Ensemble approach demonstrated better performance in classifying ticket price categories than a separate decision tree.

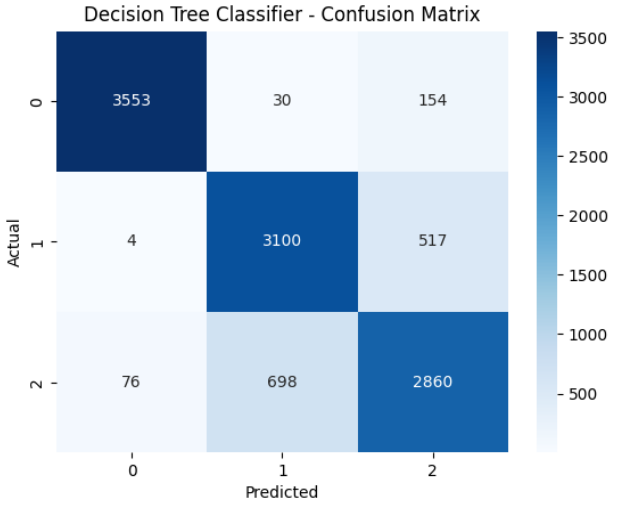

Fig. 2. Confusion Matrix for Decision Tree

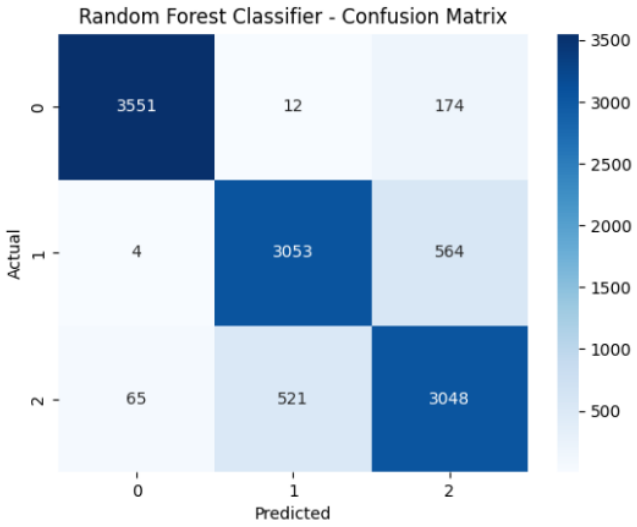

Fig. 3. Confusion Matrix for Random Forest

VI. DISCUSSION

The results obtained during the experiments show the different performance between the decision tree and the Random Forest method. Although both models work on the same principle of partitioning the space by decision logic, the difference in their performance is due to how they approach variability and robustness of classification.

The decision tree achieved an accuracy of 86.54%, which is quite respectable for a model trained on discretely categorized data. However, due to its deterministic nature, it is prone to overfitting, especially with complex or imbalanced data. Moreover, the tree can be very sensitive to small changes in the training data, which can lead to unpredictable generalization behavior.

In contrast, Random Forest as an ensemble method mitigates this problem. It combines the predictions of multiple trees that

are trained on randomly bootstrapped samples with random subsets of attributes. This stochasticity increases the diversity of the individual trees and reduces the variance of the resulting model, which is reflected in the higher accuracy of 87.81%. The accuracy improvement of 12.19% indicates that Random Forest coped better with the internal structure of the data and was also able to better recognize borderline cases.

Confusion matrices further revealed that Random Forest significantly reduced the number of confusions between the medium and high classes, which were a frequent source of error for the decision tree. This improvement demonstrates Random Forest's ability to better handle blurred decision boundaries between classes with similar characteristics.

However, from a practical point of view, computational complexity must also be taken into account. Random Forest is a more complex model - training multiple trees and aggregating their outputs requires more memory and computational time. In resource-constrained environments, it may therefore be more appropriate to use an optimized or pruned decision tree.

The results also show that even a simple classifier can achieve reasonable performance if properly designed and trained. The combination of discretely categorized inputs and a robust implementation of entropy partitioning enabled efficient modeling even without advanced tools.

VII. Conclusion

In this paper, we discuss the design, implementation and comparison of two classification models: the Decision Tree and the Random Forest methods. Both models were implemented from scratch in Python without the use of off-the-shelf classifiers, thus ensuring transparency of computation and complete control over the learning logic.

The models were applied to the task of predicting the price category of airline tickets based on several attributes. After preprocessing and categorizing the input data, the models were trained and evaluated on real data. The results showed that Random Forest achieves higher accuracy than the standalone decision tree (86.54% versus 87.81%). The difference is mainly due to the robustness of Random Forest to overlearning and its ability to combine the predictions of multiple models.

At the same time, the practical applicability of decision trees as interpretable and efficient models in tasks where it is important to understand the decision process is confirmed. Random Forest, in turn, was advantageous in situations where accuracy is more important than interpretability.

Acknowledgment

References

[1] M. Polakovič and J. Kollár, "Diskrétna matematika". 2019. [Online]. Available: https://www.math.sk/jmkollar/.

[2] T. M. Mitchell, "Machine Learning". New York: McGraw-Hill Science/Engineering/Math, 1997. ISBN: 0070428077. [Online]. Available: https://www.cs.cmu.edu/~tom/files/MachineLearningTomMitchell.pdf.

[3] T. Hastie, R. Tibshirani, and J. Friedman, "The Elements of Statistical Learning". [Online]. Available: https://www.sas.upenn.edu/~fdiebold/NoHesitations/BookAdvanced.pdf.

[4] J. Murel and E. Kavlakoglu, "What is a confusion matrix? " [Online]. Available: https://www.ibm.com/think/topics/confusion-matrix.