University of Science
Vietnam National University, Ho Chi Minh City

Report Lab 2

Artificial Intellgence

Nguyễn Hiển Đạt

Class: 21CLC06

ID: 21127591

Ho Chi Minh City, August 2023

Contents

1.1) Preparing the data sets

First read a CSV file called nursery.data.csv using the pandas library and assigns column names to the data. The data is then split into features X and labels y. Then creates training and test sets for different proportions of the data using the train_test_split function from the sklearn library. The proportions used are [0.4, 0.6, 0.8, 0.9]. The training and test sets are stored in a dictionary called subsets.

Then define a function called plot_class_distribution that takes in data and a title as arguments and creates a bar plot of the class distribution of the data. After that use this function to create plots of the class distribution of the original data set as well as the training and test sets for each proportion. These plots are displayed using the show function from the matplotlib.pyplot library.

1.2) Building the decision tree classifiers

To draw the decision tree, I use the DecisionTreeClassifier from the sklearn library to create decision trees for the different proportions of the data. First encode the categorical features in X using the get_dummies function from the pandas library. The encoded data is then split into training and test sets for each proportion using the train_test_split function from the sklearn library. The training and test sets are stored in a dictionary called subsets.

Then iterate over each proportion and creates a decision tree classifier using the DecisionTreeClassifier function with the criterion set to "entropy". The classifier is fit on the training data for that proportion. The decision tree is then visualized using the export_graphviz function from the sklearn.tree library and the graphviz library. The visualizations are saved as pdf files with format "decision_tree_{p}", where {p} is replaced by the proportion value.

1.3) Evaluate the decision tree classifiers

a) Function explanation

I use the classification_report and confusion_matrix functions from the sklearn.metrics library to evaluate the performance of the decision tree classifiers created for the different proportions of the data. The code iterates over each proportion and creates a decision tree classifier using the DecisionTreeClassifier function with the criterion set to "entropy". The classifier is fit on the training data for that proportion and used to make predictions on the test data. The predictions are then compared to the true labels using the classification_report and confusion_matrix functions. The classification report and confusion matrix are printed for each proportion.

b) Classification report and confusion matrix interpretation

The classification report provides several key metrics for evaluating the performance of a classifier, including precision, recall, f1-score, and support for each class. Precision is the ratio of true positive predictions to the total number of positive predictions. Recall is the ratio of true positive predictions to the total number of positive instances. The f1-score is the harmonic mean of precision and recall. Support is the number of instances in each class.

The confusion matrix provides a visual representation of the classifier's performance. Each row represents an actual class and each column represents a predicted class. The diagonal elements represent the number of instances that were correctly classified, while off-diagonal elements represent misclassifications.

c) Comments on the performance of decision tree classifiers

The overall accuracy is very high for all proportions, ranging from 99% to 100%.

For the proportion of 0.4, the classifier achieves perfect precision and recall for the not_recom class, indicating that all instances of this class were correctly classified. The precision and recall for the priority and spec_prior classes are also very high, at 98%. However, the classifier fails to correctly classify any instances of the recommend class, resulting in a precision and recall of 0 for this class. This is likely due to the fact that there is only one instance of this class in the test set, making it difficult for the classifier to accurately predict this class.

For the proportions of 0.6, 0.8, and 0.9, the classifier achieves perfect or near-perfect precision and recall for all classes. This indicates that the classifier is accurately predicting the classes of most instances in these test sets.

The confusion matrices provide further insight into the classifier's performance. For all proportions, the diagonal elements are large, indicating that most instances were correctly classified. The non-diagonal elements are small, indicating that there were few misclassifications.

Overall, these results suggest that decision tree classifiers perform very well on the Nursery dataset for all train/test proportions. The classifier is able to accurately predict the classes of most instances, with high precision and recall for all classes (except for the recommend class in the proportion 0.4 test set).

1.4) The depth and accuracy of a decision tree

a) Function explanation

I use the DecisionTreeClassifier from the sklearn.tree library to create decision trees for different values of the max_depth parameter.First retrieve the training and test sets for a proportion of 0.8 from the subsets dictionary. The values of max_depth used are [None, 2, 3, 4, 5, 6, 7]. Then iterate over each value of max_depth and create a decision tree classifier using the DecisionTreeClassifier function with the criterion set to "entropy" and the max_depth parameter set to the current value of max_depth. The classifier is fit on the training data and used to make predictions on the test data. The accuracy of the predictions is calculated using the accuracy_score function from the sklearn.metrics library and stored in a dictionary called accuracy_scores.

Next visualize each decision tree using the export_graphviz function from the sklearn.tree library and the graphviz library. The visualizations are saved as files with names in the format "decision_tree_{max_depth}", where {max_depth} is replaced by the value of max_depth.

Finally, print the accuracy scores for each value of max_depth.

b) Statiscial results

Max_depth	None	2	3	4	5	6	7
Accuracy	0.9961	0.7635	0.8098	0.8488	0.8727	0.8854	0.9097

c) Comment on the above statistics

When max_depth is set to None, meaning that the tree is grown to its maximum depth, the classifier achieves an accuracy of 99.61% on the test set.

When max_depth is set to a smaller value, such as 2 or 3, the classifier's accuracy decreases significantly, to 76.35% and 80.98%, respectively. This suggests that a shallow decision tree with a small maximum depth is not able to accurately capture the relationships between the features and the target variable in this dataset.

As max_depth is increased from 4 to 7, the classifier's accuracy gradually improves, reaching a maximum of 90.97% when max_depth is set to 7. This indicates that allowing the decision tree to grow deeper and fit more complex decision boundaries can improve its classification accuracy on this dataset.

Overall, the maximum depth of the decision tree can significantly affects its classification accuracy on the Nursery dataset. Allowing the tree to grow deeper can improve its accuracy, and setting max_depth too low can result in a significant decrease in performance.