UCLouvain-EPL

Mining Patterns in Data

Project 3: Classifying Graphs

Teacher:
Siegfried Nijssen
Course assistants:
Thomas Charles
Guillaume Derval

Authors:
Nima Farnoodian - 68372000
nima.farnoodian@student.uclouvain.be

César Liesens - 39161400
cesar.liesens@student.uclouvain.be

Group 53

May 14, 2021

1 Introduction

In this project, we trained different classifiers for the molecules datasets using the Pattern-based classification approaches. The dataset contains graphs, which represent the molecules. This project aims to extract the top-k frequent subgraphs that have high confidence from two datasets (mutagen (positive) and non-mutagen (negative)) in order to train classifiers to predict the mutagenicity of the molecules. To this end, we used the gSpan algorithm to extract the subgraphs based on which the classifiers are trained. Since the gSpan algorithm was beforehand given, our effort mostly focused on defining different tasks on gSpan to extract subgraphs and train different classifiers as requested.

In what follows, in section 2, we will discuss the strategies we applied for detecting the subgraphs with high confidence and the way that we extended the provided implementation. Finally, in section 3, we will present our experimental results in-depth.

2 Strategies

In this section, we will briefly converse about the chosen strategies for the given tasks.

2.1 Top-k Mining with gSpan

Top-k Mining with gSpan aims at extracting the top-k subgraphs with high confidence on the positive class and whose frequencies are above a user-specified threshold. To hold top-k subgraphs, we initially applied the data structure called "K-selector" that we utilized in the previous project. K-selector has an append method that appends a pattern with its score (e.g., support, confidence, wracc, etc.) to a data structure that keeps only top-k patterns where "top-k" is defined based on the given score. K-selector was designed in such a way that it ensures the repetition of the patterns with the same score. Using this data structure, we could store all top-k patterns with high confidence. Since the project asks us to consider supports of the patterns if there are many patterns with the same confidence, we applied a post-processing phase to remove the patterns with low supports if their confidences are the same. Indeed, in the post-processing, we used a k-selector data structure for each confidence value to only select the pattern(s) with the highest support if there are many, or all patterns with the same confidence and supports if there is not any high frequent pattern.

Although in the previous project, K-selector has proved efficient in pruning search trees and could guarantee that only top-k patterns are selected, the output of our program was rejected by the ingenious test for the large dataset. We still do not know why the error happened since we checked the output of the algorithm several times and compared it with other samples. However, after three days working to find out the error, we made a decision to use a naive solution for the Top-k Mining problem. In our naive solution, the algorithm first store all detected patterns whose supports are above a threshold. We defined an object named "pattern" that holds the dfs_code, confidence, support, and gid_subsets for a pattern x. When finding a pattern, the algorithm creates "a pattern object" for x, and appends the object to a list like the given prototype. After detecting all frequent patterns, in the post-processing, it sorts the pattern list by the descending multiple levels of sorting (to sort by confidence then by support). Given that task.patterns refers to the pattern list, the following python code sorts the pattern list as explained. After sorting, the algorithm selects the patterns sequentially until k patterns are selected —select the pattern x if only x has the confidence z, otherwise select the highly frequent pattern with the confidence z or select all frequent patterns with z.

```
from operator import attrgetter
sorted(task.patterns, key=attrgetter('confidence', 'support'), reverse=True)
```

2.2 Training a basic model

In this task, we were asked to train a decision tree classifier to predict the mutagenicity of the molecules. The classifier must be trained according to the top-k frequent patterns with high confidence. The problem of top-k patterns is here analogous to 2.1. To provide a better estimate of the performance of the classifier, we used the stratified k-fold cross-validation method to monitor the classifier performance on each fold and realize the variation of the accuracy on each test set. To implement this task, we obeyed the prototype with a modification. The datasets (both positive and negative datasets) are split into k folds. At each fold, the model is trained based on the k-1 folds (train set) and then tested according to the leave-out set (valid set). During each fold, the top-k frequent patterns with high confidence in the train set are detected as defined in 2.1. Then, two tabular data frames (one for the training set and one for the valid set) are created, whose columns represent the detected patterns, and rows are the transactions in the training and valid datasets. In these data frames, attributes(patterns) are 1 if they exist in the related transactions otherwise they are assigned 0. Next, a decision tree is trained using the training data frame and is tested based on the valid data frame. Generally speaking, this task is similar to the one provided in the prototype, but this one needs to extract the top-k subgraphs and train a decision model using found subgraphs.

2.3 Another classifier

Generally, this task is an extension of 2.2. In this task, we are allowed to try different classifiers, play with tuning parameters, and even change mining task (e.g, different k or support thresholds). For training the data, we picked ensemble learning techniques such as Voting Classifier as we found single classifiers less reliable. The aim of using the ensemble learning techniques was to reduce the variance as the single classifiers like Decision Tree, Random Forest, etc., were prone to over-fitting on the test dataset. The chosen single classifiers are Multi-layer Perceptron (MLP), Support Vector Machine (SVC), and Random Forest (RF). The approach that we used to build Voting Classifier is as follows. For every single classifier, we initially tried to find the best configuration using cross-validation. We found out that changing the hyper-parameters of the classifiers does not lead to any significant improvement in the outcome of the voting classifier. Instead, pattern mining-related parameters such as k and support threshold can significantly change the expected accuracy and also the variance. By this, we mean that even the default configuration of the single classifiers could give us the same accuracy as the best configuration, but in contrast, pattern mining-related parameters could define a new set of patterns thereby a different trained classifier. Therefore, we fed the Voting Classifier with the default configuration of the single models. The voting type is soft, meaning the prediction is accomplished based on the average of the probabilities returned by classifiers. Our best setting for the pattern mining task is k=15 and minsup=0.2. We found these two parameters using grid search with 4 folds. In section 3, we will illustrate the impact of these two parameters on our voting classifier, and explain why we have chosen them. In what follows, you will see the configuration of our classifier:

```
MLP=MLPClassifier(random_state=1, max_iter=500)
svc=SVC(probability=True)
RF=RandomForestClassifier(max_depth=2, random_state=0)

classifier = VotingClassifier(estimators=[('mlp', MLP),
('svc', svc),
('rf',RF)], voting='soft')
```

3 Performance Evaluation

In this section, we compare the performance of the classifiers and also show the effect of pattern-mining parameters on them. Here, we also discuss why we have chosen k=15 and minsup=0.2 for our submission on ingenious.

The molecule datasets are listed in Table1 in ascending order of magnitude. The implementations, measurements, and comparisons were carried out in Python 3.8.3 under Windows 10 on a Dell all-in-one PC powered by Intel Core i5-4590S CPU @ 3.00GHz and 8 GB of main memory.

	# of Transactions in positive dataset	# of Transactions in negative dataset
Medium Molecule Dataset	153	137
Molecule Dataset	1801	1452

Table 1: Sample Transaction Datasets

3.1 Label Prediction based on detected patterns

To see how different patterns may lead to a supervised classifying model, we ran a grid search over the classifiers with 4 folds and different k and minsups (k,minsup). Figures 1 and 2 show the results on the medium and large datasets, respectively. On the medium size, we cannot infer anything meaningful regarding the parameters since the size of the dataset is not large. Nevertheless, one observation in Figure 1 indicates that any combination of k with minsup=0.2 results in higher expected accuracy with a bit higher standard deviation in both classifiers. Surprisingly, although the voting classifier is more complex and expected to provide better results, no such improvement is seen in both medium and small datasets. Indeed, the classifiers may not play a critical role in the prediction accuracy but do the pattern mining-related parameters. As seen in Figure 2, the best accuracy belongs to k=15 and minsup=0.2. Moreover, the standard deviation is relatively low for these parameters, indicating a promising variance on an unseen dataset. As these two parameters gave us the best results for our classifier, we suggested our ensemble voting model with these parameters in 2.2.

In conclusion, the pattern mining-based machine learning task is challenging and requires so much effort in order to build a reliable model. Training a machine learning model, in turn, needs parameter tuning, performance evaluation, and more importantly a finalized dataset —we should reach a consensus concerning our dataset obtained from the extracted patterns. Obtaining such a dataset may involve so much uncertainty: what scoring function should be used? what is the support threshold? How to define the top-k problem? and so on. All these possibilities make the pattern mining-based machine learning task non-trivial and an iterative process. Thereupon, the model that we submitted may get ameliorated by changing a simple parameter that was concealed from our eyes. We tried to find a good model but we ended up with the fact that minsup and top-k should be tuned.

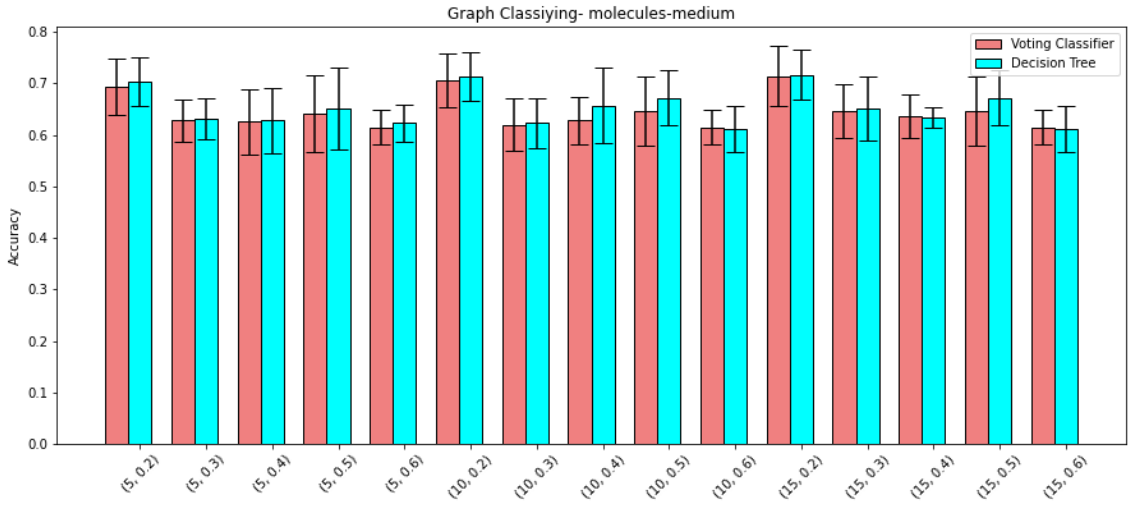

Figure 1: Dataset: Molecules-medium. Average Accuracy. The error bar on bar-tip indicates Standard Deviation

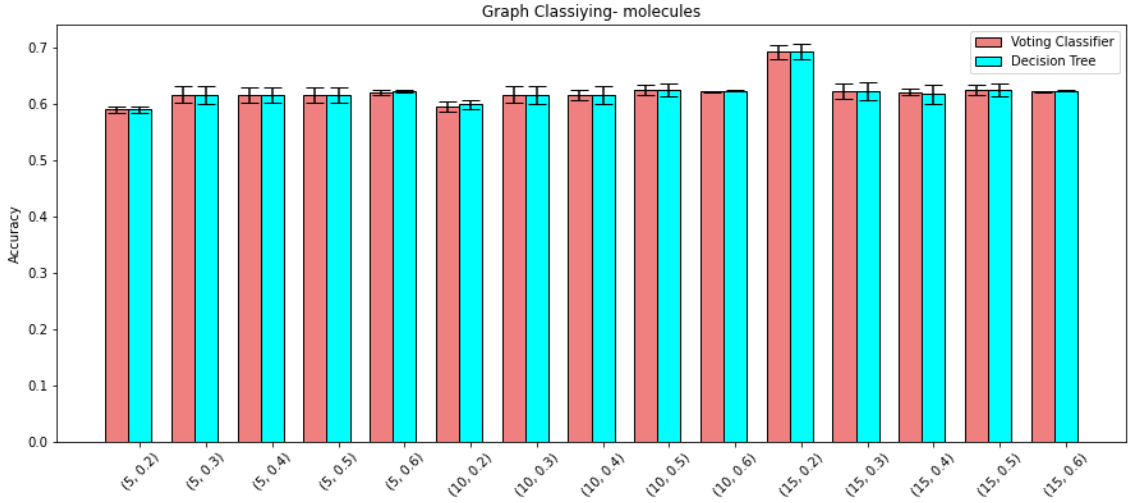

Figure 2: Dataset: Molecules. Average Accuracy. The error bar on bar-tip indicates Standard Deviation