Coursework Distributed Databases

Jens De Staercke Edouard Goddeeris Arno Temmerman

December 5, 2021

Abstract

This is our report of the assignment for the course "Distributed Databases". In the following chapters we will describe how we applied the automated machine learning toolkit auto-sklearn to multiple machine learning problems.

Contents

Chapter 1

Excercise Mammography

As an introduction to the auto-sklearn toolkit we applied it to one of the exercises given in our course.

In the exercise Mammography we originally predicted the severity (benign/malign) of mammography results, with the use of a RandomForestClassifier. This time, however, we will be using the AutoSklearnClassifier and find out if we can get any useful predictions.

1.1 Setting up the environment

As recommended, we created a Google Colab notebook. The installation of auto-sklearn can be done with the following command:

[ ]:
```
!sudo apt-get install build-essential swig python3-dev
!pip3 install auto-sklearn
```

After this, restart the runtime (Ctrl + M). You can now import the module:

[ ]:
```
import autosklearn
autosklearn.__version__
```

1.2 Importing and cleaning the data

Importing and cleaning the data could be done in the exact same way of the original exercise, using pandas (short version):

[ ]:
```
url = 'https://raw.githubusercontent.com/HOGENT-Databases/DB3-Workshops/
 ↪master/data/mammography.csv'
mammo = pd.read_csv(url)

# Remove all lines that contain some "?".
has_missing = (mammo == "?").sum(axis=1).astype('bool')
mammo = mammo[~has_missing]
```

```
# Remove any observation that has an incorrect value in the BIRADS␣
 ↪column.
valid_rows = mammo['BIRADS'].isin(['0', '1', '2', '3', '4', '5'])
mammo = mammo[valid_rows]
```

```
Shape: (820, 6)
```

[ ]:
```
BIRADS Age Shape Margin Density  Severity
0       5  67     3      5        3         1
2       5  58     4      5        3         1
3       4  28     1      1        3         0
8       5  57     1      5        3         1
10      5  76     1      4        3         1
```

1.3 Determining a model with auto-sklearn

First we'll seperate the input(X) from the output data(y).

[ ]:
```
# Step 1: Set up the required DataFrames
COLUMNS = ['Age', 'Shape', 'Margin', 'Density']

mammo[COLUMNS] = mammo[COLUMNS].apply(pd.to_numeric)

X = mammo[COLUMNS]
y = mammo['Severity']
```

```
X:
Age  Shape  Margin  Density
0    67      3       5        3
2    58      4       5        3
3    28      1       1        3
8    57      1       5        3
10   76      1       4        3

y:
0     1
2     1
3     0
8     1
10    1
```

Next, we'll split the data into a training, a validation and a test set.

[ ]:
```
# Step 2: Split the data into training, validation and test set
from sklearn.model_selection import train_test_split

X_rem, X_test, y_rem, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)
```

```
X_train, X_val, y_train, y_val = train_test_split(X_rem, y_rem,␣
↪test_size=0.3)
```

```
Number of training examples: 401, number of features 4
Number of validation examples: 173
Number of test examples: 246
```

Now we can build the model with the use of auto-sklearn. Since this is a binary classification problem (the BI-RADS data should be classified as either benign or malignant), we can make use of the AutoSklearnClassifier:

[ ]:
```
# Step 3. Build the classifier
import autosklearn.classification
from sklearn.metrics import accuracy_score

model = autosklearn.classification.AutoSklearnClassifier(
        time_left_for_this_task=120,
        per_run_time_limit=30,
)
model.fit(X_train, y_train)
```

It's very important to specify the following parameters:

- **time_left_for_this_task:** time (seconds) in which auto-sklearn can search for the appropriate models.
- **per_run_time_limit:** time (seconds) auto-sklearn gets for fitting the model on the training data.

Part of our job during this assesment, was to fiddle arround with these parameters, in order to get better and better models/predictions: Ascribe large values and you increase the odds of finding a model better fitted to the (training) data. But it will take a long time to get to this model (with the risk of overfitting). Ascribe lower values to save time, with the risk of ending up with a bad model.

At the very beginning of this assignment, we completely forgot to specify these parameters. As a result the process took so long, that we never got it to finish. Once the paramaters were set, we received a reasonably good model right away.

We can now use the validation set to check wether or not this model can give us any reasonable predictions:

[ ]:
```
# Step 4. use the model on the validation set
y_val_pred = model.predict(X_val)

# Compare the predictions of the model with the actual Severity
accuracy = accuracy_score(y_val, y_val_pred)
```

```
Accuracy of 0.8034682080924855
```

This accuracy score gives us confidence that this model is actually useful for making predictions. We feel ready to use it on the test set:

[ ]:
```
# Step 5. use the model on the test set
y_test_pred = model.predict(X_test)

# Compare the predictions of the model with the actual Severity
test_accuracy = accuracy_score(y_test, y_test_pred)
```

```
Estimated accuracy of the model is 0.8048780487804879
```

Our model would have correctly diagnosed about 80% of the instances in the test set. (Which is actually better than what the random forest classifier in the original exercise could manage.)

Chapter 2

Kaggle competitions

For this part of the assignment, we chose Investigate the models created by auto-sklearn and compare them with the winning models of the Kaggle competitions. Did you get a comparable performance? Do you get better models if you increase the "time" budget? What type of model is the winning model? What types of models does auto-sklearn use? Compare the required effort and required domain knowledge between the winning models and the models created by auto-sklearn.

2.1 Binary classification

Random acts of pizza

2.2 Multiclass classification

Forest cover type prediction
Abstract classification

2.3 Regression

House prices
Bike sharing