Glass Classification

Patrick Bettermann

16.12.2020

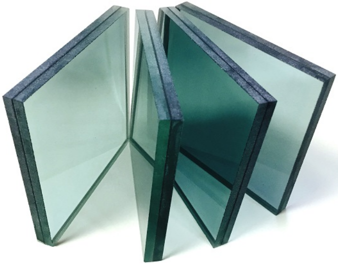
Is it used for?

- buildings?
- cars?
- tableware?

Furthermore

is it processed in a special way?

What type of glass is this?



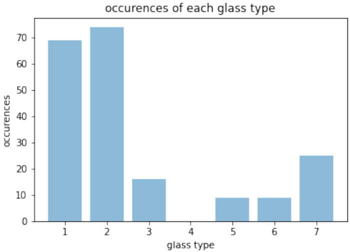
Dataset

- `https://www.kaggle.com/uciml/glass`
- only 214 entries of different types (7) of glass
- glass characterized by
  - RI: refractive index
  - Na: Sodium (weight percent in corresponding oxide)
  - Si: Silicon

  ...
- which are therefore correlated

Prepare data

- input: csv file → .txt
- 70% of the entries are type 1 and 2

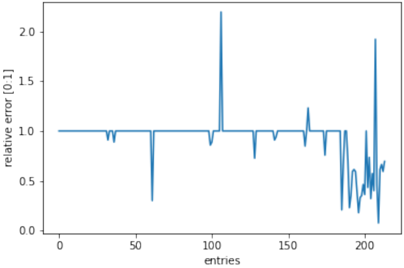


- cleaning: delete entrie if value differs by $\pm\bar{x}$
  - zero values not counted

Prepare data - cleaning

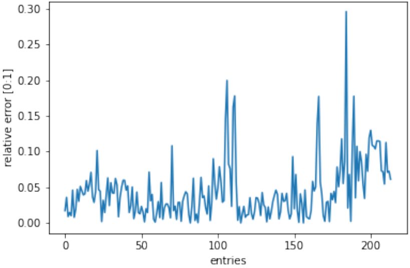
Ba: Barium

- 176 of 214 values empty
- some outliers
- mostly present in type 7



Na: Natrium

- no outliers
- present in all types of glass



Random Forests

- reliant model
- key ideas:
  - randomnes of the tree construction
  - many randomly constructed trees should compensate for errors
  - random and independant data selection
- can overfit

Encoding

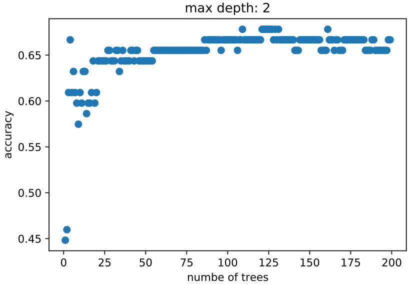
- no encoding needed
- only seperation of data and labels is performed

Tuning

- parameters: max. depth, number of trees, how to split the tree, ...

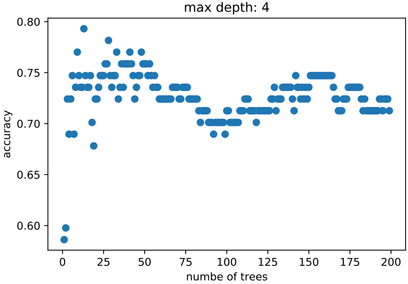
max. depth = 2

- capped at 65%

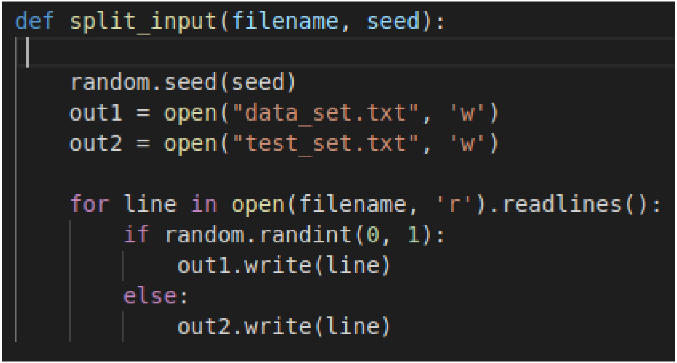


max. depth = 4

- up to 79% accuracy



Train/Validation



```python
def split_input(filename, seed):

    random.seed(seed)
    out1 = open("data_set.txt", 'w')
    out2 = open("test_set.txt", 'w')

    for line in open(filename, 'r').readlines():
        if random.randint(0, 1):
            out1.write(line)
        else:
            out2.write(line)
```

- problem: data set is very small
  now it's even smaller

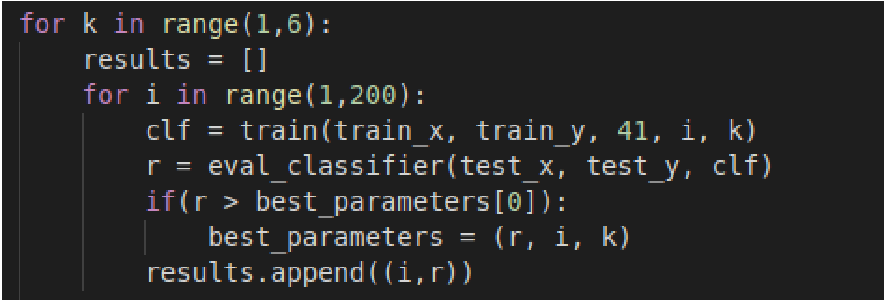
Results

- best parameters:
  number of trees = 13
  max. depth = 4
  → result: 79% of predictions are correct
- only 200 entries → trees don't need to be complex

Discussion

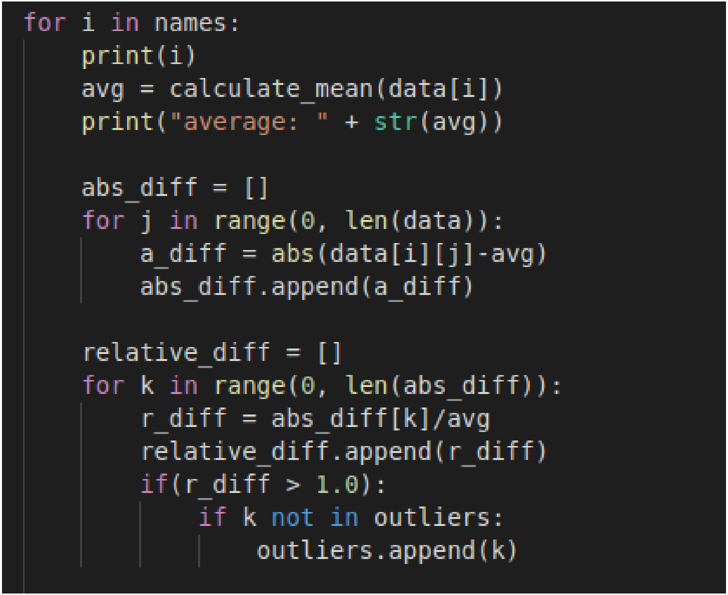
- check for overfitting is missing
- how to handle zero values?
- Are there better methods to find outliers than difference to mean?
- Maybe leave-one-out-cross-validation might work since the data set is so small?

Code - evaluation



```python
for k in range(1,6):
    results = []
    for i in range(1,200):
        clf = train(train_x, train_y, 41, i, k)
        r = eval_classifier(test_x, test_y, clf)
        if(r > best_parameters[0]):
            best_parameters = (r, i, k)
        results.append((i,r))
```

Code - get an idea of the data



```
for i in names:
    print(i)
    avg = calculate_mean(data[i])
    print("average: " + str(avg))

    abs_diff = []
    for j in range(0, len(data)):
        a_diff = abs(data[i][j]-avg)
        abs_diff.append(a_diff)

    relative_diff = []
    for k in range(0, len(abs_diff)):
        r_diff = abs_diff[k]/avg
        relative_diff.append(r_diff)
        if(r_diff > 1.0):
            if k not in outliers:
                outliers.append(k)
```