Lazy FCA Report

Prikhno M.A. 13.12.2022

Dataset

Body signal of smoking:
https://www.kaggle.com/datasets/kukuroo3/body-signal-of-smoking

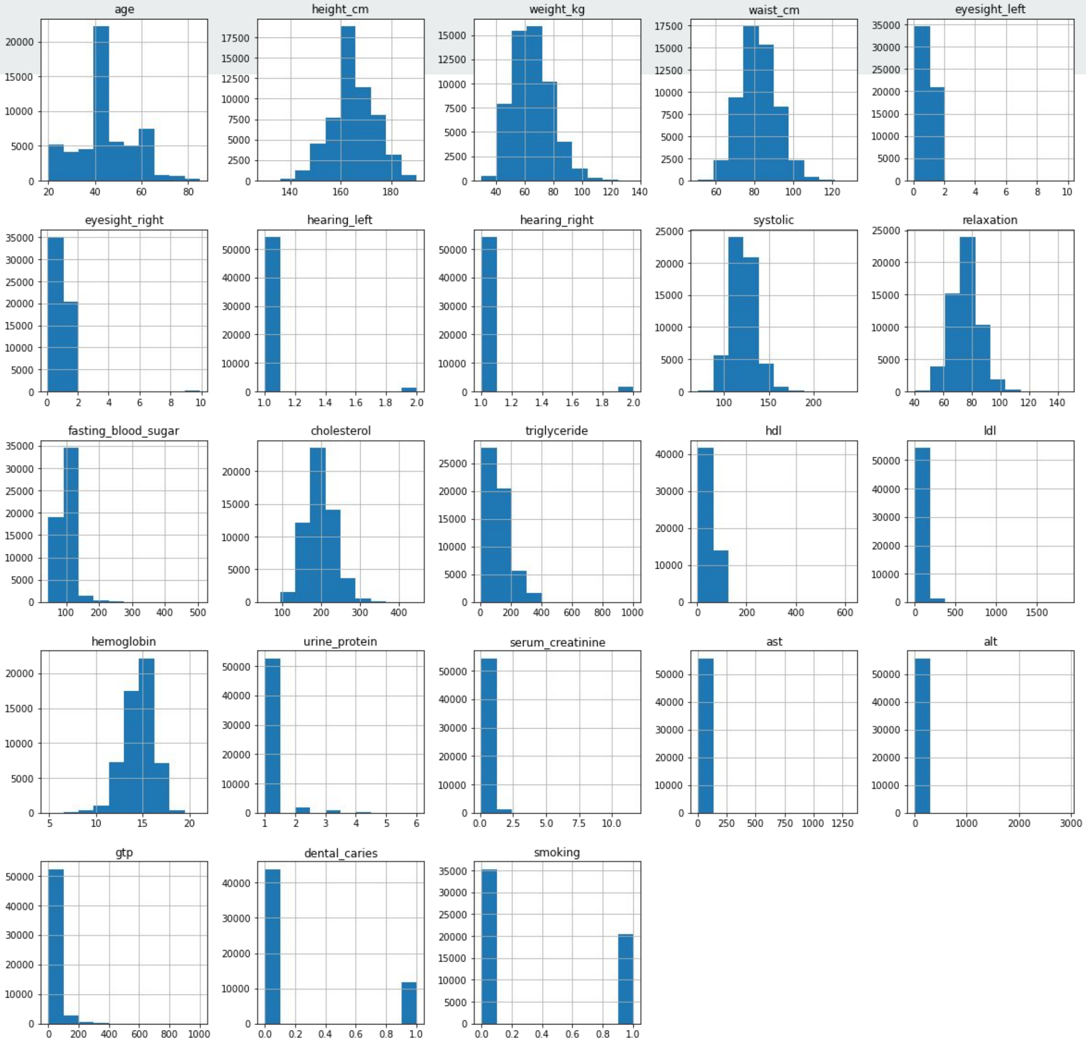
Target: find smokers based on their vital signs and medical data

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 55692 entries, 0 to 55691
Data columns (total 26 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   gender               55692 non-null  object
 1   age                  55692 non-null  int64
 2   height_cm            55692 non-null  int64
 3   weight_kg            55692 non-null  int64
 4   waist_cm             55692 non-null  float64
 5   eyesight_left        55692 non-null  float64
 6   eyesight_right       55692 non-null  float64
 7   hearing_left         55692 non-null  float64
 8   hearing_right        55692 non-null  float64
 9   systolic             55692 non-null  float64
 10  relaxation           55692 non-null  float64
 11  fasting_blood_sugar  55692 non-null  float64
 12  cholesterol          55692 non-null  float64
 13  triglyceride         55692 non-null  float64
 14  hdl                  55692 non-null  float64
 15  ldl                  55692 non-null  float64
 16  hemoglobin           55692 non-null  float64
 17  urine_protein        55692 non-null  float64
 18  serum_creatinine     55692 non-null  float64
 19  ast                  55692 non-null  float64
 20  alt                  55692 non-null  float64
 21  gtp                  55692 non-null  float64
 22  oral                 55692 non-null  object
 23  dental_caries        55692 non-null  int64
 24  tartar               55692 non-null  object
 25  smoking              55692 non-null  int64
dtypes: float64(18), int64(5), object(3)
memory usage: 11.0+ MB
```

Dataset

- categoric values are converted to indicator values via pd.get_dummies()
- continuous values are first converted to categorical via pd.qcut() method that assigns labels to values according to bins, and then to indicator values via pd.get_dummies()

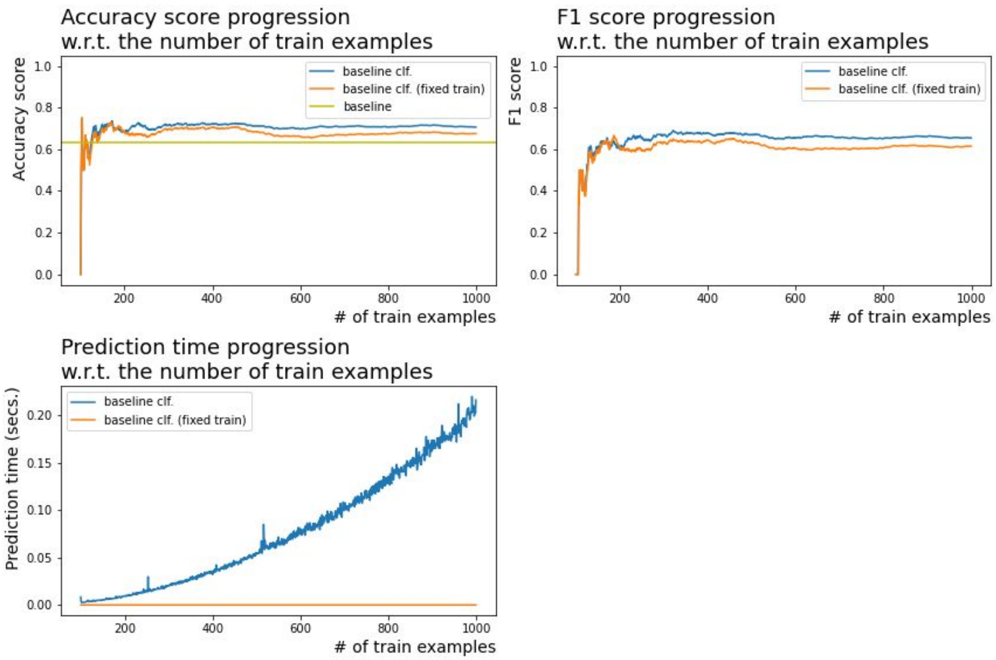


Quality measure

Let's use two metrics: accuracy score and F1 score.

- accuracy score is a valid metric, because the dataset is balanced (there is no clear imbalance between "non-smokers" and "smokers")
- F1 score minimizes the False Negative prediction which is the most harmful in the case of our dataset (when someone is a smoker but was not detected as such)

Original prediction function



Time: 1min 9s with train updates and 2.28 s without train updates

Numpy-modified prediction function

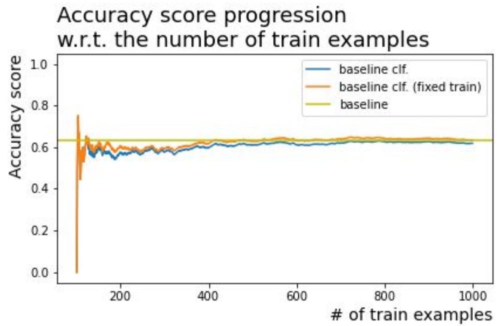
Principle:

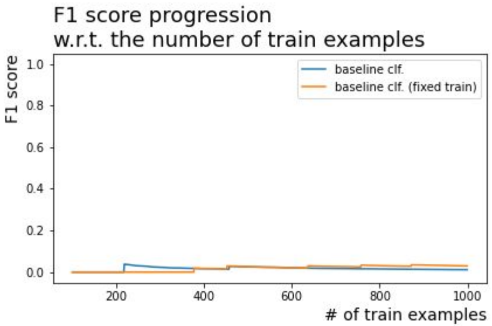
1. List[set] -> 2D np.array
2. Loop through rows in X_pos and calculate intersections_pos = x & row of X_pos -> x.reshape(1, -1) & X_pos
3. Length of intersection >= min_cardinality -> intersection.sum() >= min_cardinality
4. Count X_neg that contain intersection -> calculate zeros in the product of the intersection and negated transposed X_neg

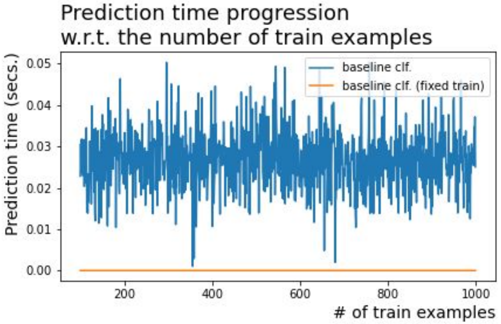
example for principle 4:

intersec (0, 0, 1, 0);	x_neg (0, 1, 0, 1);	~x_neg (1, 0, 1, 0)	->	intersec @ (~x_neg.T) = 1	->	not contained
intersec (0, 0, 1, 0);	x_neg (0, 1, 1, 1);	~x_neg (1, 0, 0, 0)	->	intersec @ (~x_neg.T) = 0	->	contained
intersec (1, 0, 1, 0);	x_neg (0, 1, 1, 1);	~x_neg (1, 0, 0, 0)	->	intersec @ (~x_neg.T) = 1	->	not contained

Numpy-modified prediction function







- Accuracy and F1 scores are lower than in the original algorithm
- But runtime improved significantly

Time: 24.2 s with train updates and 529 ms without train updates

Model comparison

Accuracy scores of:

DecisionTreeClassifier: 0.68 **RandomForestClassifier: 0.71** **CatBoostClassifier: 0.73**

Versus accuracy score of Lazy algorithms:

Original: 0.7063403781979978 **Modified: 0.6184649610678532**

Conclusion

- the chosen dataset of binary classification was prepared and binarized for the task
- the original lazy classifier was used for prediction with resulting scores: **accuracy score: 0.7063403781979978**, **F1 score: 0.6544502617801047**; time spent for prediction was **1min 9s** with train updates and **2.28 s** without train updates
- lazy classifier was enhanced and translated to numpy; resulting scores in prediction: **accuracy score: 0.6184649610678532**, **F1 score: 0.011527377521613834**; time spent for predictions: **24.2 s** with train updates and **529 ms** without train updates
- popular rule-based models were used for prediction with resulting accuracy scores: **DecisionTreeClassifier: 0.68**, **RandomForestClassifier: 0.71**, **CatBoostClassifier: 0.73**