Data

UCI "Heart Disease" was used for training and testing the algorithm. It contains 14 attributes describing health condition of 303 different patients. The "target" attribute used for classification refers to the presence of heart disease in the patient. First five objects of the dataset are shown in the table below.

In [103]:

```
df.head()
```

Out[103]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	tar
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

As one can see, the dataset contains non-binary features. Since FCA-based classification approaches require binary data, non-binary features were binarized. Numerical features were distributed across 5 equally spaced bins. Categorial features were binarized using one-hot encoding. The resulting dataset has 50 features.

In [107]:

```
X.head()
```

Out[107]:

	age_(28.952, 38.6]	age_(38.6, 48.2]	age_(48.2, 57.8]	age_(57.8, 67.4]	age_(67.4, 77.0]	sex_(-0.001, 0.5]	sex_(0.5, 1.0]	cp_(-0.0 0.
0	0	0	0	1	0	0	1	
1	1	0	0	0	0	0	1	
2	0	1	0	0	0	1	0	
3	0	0	1	0	0	0	1	
4	0	0	1	0	0	1	0	

5 rows × 50 columns

Algorithm

Original context is split into positive and negative part. The description of the classified object from the test sample is compared with the description of each example from the positive context. If the sum of their intersection crosses the threshold, then this example from the positive context "votes" for the classified object.

$$|g' \cap g'_+| <> |g'| \cdot threshold$$

Similar comparisons are made with examples from the negative context. Further, the sums of votes are normalized by the number of examples in +/- contexts. Then normalized votes are used for classification. Python3 implementation of this algorithm is shown below.

In [101]:

```
class FCAClassifier(BaseEstimator, ClassifierMixin):

    def __init__(self, threshold=0.5):
        self.threshold = threshold

    def fit(self, X, y):

        # Check that X and y have correct shape
        X, y = check_X_y(X, y)
        # Store the classes seen during fit
        self.classes_ = unique_labels(y)

        self.X_ = X
        self.y_ = y

        #Split context into positive and negative parts
        self.X_pos_ = X[y == 1]
        self.X_neg_ = X[y == 0]

        # Return the classifier
        return self

    def predict(self, X):

        # Input validation
        X = check_array(X)

        y_pred = []

        # Classification algorithm
        for obj in X:
            pos = 0
            neg = 0
            for pos_obj in self.X_pos_:
                if np.sum(obj == pos_obj) > int(len(pos_obj) * self.threshold):
                    pos += 1
            for neg_obj in self.X_neg_:
                if np.sum(obj == neg_obj) > int(len(neg_obj) * self.threshold):
                    neg += 1

            pos = pos / len(self.X_pos_)
            neg = neg / len(self.X_neg_)
            if (pos > neg):
                y_pred.append(1)
            else:
                y_pred.append(0)

        y_pred = np.array(y_pred)

        return y_pred

    def score(self, X, y):
        y_pred = self.predict(X)
        return accuracy_score(y, y_pred)
```

Tuning the hyper-parameter

Baseline accuracy of the FCA-based classifier with default threshold value of 0.5 appeared to be very low (0.51).Threshold parameter was tuned by performing exhaustive grid search over [0.1, 1) interval. During the search it was found that threshold=0.7 maximizes the accuracy score. As a result, classifier's score increased up to 0.83.

Performance

We use k-fold cross-validation with k=10 and a set of 8 metrics to estimate the performance of our FCA-based algorithm. The averages of the computed values are shown in the table below. The results are compared with two popular classification algorithms: k-nearest neighbors algorithm and logistic regression.

Metric	KNN	Logistic regression	FCA-based
accuracy	0.788	0.838	0.831
precision	0.767	0.844	0.808
recall	0.884	0.872	0.914
F1	0.819	0.855	0.856
true positive	146	144	151
false positive	45	28	37
true negative	93	110	101
false negative	19	21	14

Conclusion

As the table above indicates, FCA-based algorithm performs noticably better than k-nearest neighbors algorithm and slightly worse than logistic regression depending on the metric of interest. In general, after threshold parameter was tuned, the developed algorithm managed to show decent performance on the real world data. The proposed approach can be further improved by experimenting with different characteristics used for classification.