Lazy FCA Report
Ordered sets

By PRIYANKA SINGH

Introduction

This Lazy FCA homework is the merger of lazy learning and Formal Concept Analysis.

It introduces three main topics:
1) Typical machine learning project: The pipeline of loading a dataset , designing a new predictive algorithm, results comparison.
2) Lazy learning : Predicting labels for small or rapidly changing data.
3) Rule-learning: Viewing data as binary descriptions of objects(instead of points in a space of real numbers).

Lazy learning

Lazy learning is a learning method in which generalization of the training data is, in theory, delayed until a query is made to the system, as opposed to eager learning, where the system tries to generalize the training data before receiving queries.

We do lazy learning as in KNN algorithm because data set is continuously updated with new entries (e.g., new items for sale at Amazon, new movies to view at Netflix, new clips at YouTube). Because of the continuous update, the "training data" would be rendered obsolete in a relatively short time especially in areas like books and movies, where new best-sellers or hit movies/music are published/released continuously. Therefore, one cannot really talk of a "training phase".

Lazy classifiers are most useful for large, continuously changing datasets with few attributes that are commonly queried.

Advantage of this system is achieved if the predictions using a single training set are only developed for few objects.

Disadvantages of this system is
1) Larger space is required to store the training data.
2) They are usually slower to evaluate.

Formal concept Analysis

FCA is a principled way of deriving a concept hierarchy from a collection of objects and their properties. Each concept in the hierarchy represents the objects sharing some set of properties; and each sub-concept in the hierarchy represents a subset of the objects (as well as a superset of the properties) in the concepts above it.

Binary Classification

We are given the data X= x1,x2,x3…… and the corresponding labels Y where each label is classified as either True or False. We have to make a prediction of label y for each datum x as if y is unknown.

To estimate the quality of predictions we split the data X into two non overlapping sets X train and Xtest. After this we make the prediction for each test datum based on the information obtained from training data Xtrain. Finally we measure how well the predictions represent the true test labels.

Original data has various forms: integers, categories etc , thus to make all these data look same we binarize the dataset.

Baseline Algorithm

Suppose we want to make prediction for dataset given the set of training examples Xtrain and labels as True or False corresponding to each data that belongs to training set.

First we split all counterexamples of training set to positive Xpos and negative Xneg.

For classification:

1) Count the number of examples for positive examples.
   For each positive example we compute the intersection of x with Xpos. Then we count the counterexamples for this intersection, that is the number of negative examples containing intersection of x with Xpos
2) Count the number of examples for negative examples.
   For each negative example we compute the intersection of x with Xneg. Then we count the counterexamples for this intersection, that is the number of positive examples containing intersection of x with Xneg.

Finally we compare the average number of counterexamples for positive and negative examples. We classify x being positive if the number of counterexamples for positive examples is smaller than the one for negative examples.

Dataset

The dataset used is heart failure clinical records. In this dataset there were 300 instances with 13 attributes and no missing value.

Attributes
1)age
2)anemia
3)creatinine phosphokinase
4)diabetes
5)ejection fraction
6)high blood pressure
7)plateletes
8)serum cretinine
9)serum sodium
10)sex
11)smoking
12) time
13)death event: The death event has value True or False.

Now we have to binarize our dataset and make every value 0 or 1.

Binarization

For binarization I have used the lazy pipeline method. For this we have to use python library pandas. Pandas offers data structure and operations for powerful, flexible, and easy-to-use data analysis and manipulation.
get_dummies() function is used for data manipulation and converts categorical data into dummy variables.

Prediction

To check the prediction, the metrics used were Accuracy and F1 score.
Accuracy measures the all correctly identified cases.
F1 score is harmonic mean of precision .

For both Accuracy and F1 score, the worst possible case is 0 and best score is 1 and this shows how the model was predicted accurately.

Pipeline Adaptation

By adapting lazy_pipeline.py first the data was uploaded using load_data() function, and this calls another function load_heart_failure_clinical_records_dataset() for loading the data.

For binarizing the the value we have taken death event as either True or False.

For binarizing the data we call the function binarize_X(). Inside this function we use pd.get_dummies() function.

After binarizing the data we shuffle the X(data) and then with the help of y(classes) orders the data to follow the rows order from X(data).

After shuffling we convert out dataset into sets(X_bin) and with the help of this splits our data into training set and test set. Here we are taking 90% of of data as training set and remaining 10% as test set.

After splitting our dataset we call our prediction function pipe.predict_array() which takes X_bin i.e set of data , y(classes) as a list , training data snd use_tqdm for boolean value.

We call the function pipe.apply_stopwatch() to show the predicted time.

For this dataset the algorithm takes 953ms to predict the classes to test samples.

To calculate the accuracy and f1 score we use python library sklearn.metrices.

After predicting time we compute prediction metrices which includes accuracy and f1 score and plot them with the help of python library matplotlib.pyplot as plt.

For this dataset our algorithm shows 68% accuracy and f1 score is 8. Accuracy shows that our algorithm is correctly identifying only 68% of our data .

Other dataset

Now we apply this algorithm to another dataset. The dataset used is a stoke dataset.

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths.

This dataset is used to predict whether a patient is likely to get stroke based on the input parameters like gender, age, various diseases, and smoking status. Each row in the data provides relavant information about the patient.

Here we predict if a person can have stroke or not with the help of some attributes. In this dataset there are 5110 observations with 12 attributes that are id, gender, age, hypertension, heart disease, ever married, work type, residence type, avg glucose level, bmi, smoking status, stroke.

Attribute Information

1) id: unique identifier

2) gender: "Male", "Female" or "Other"

3) age: age of the patient

4) hypertension: 0 if the patient doesn't have hypertension, 1 if the patient has hypertension

5) heart_disease: 0 if the patient doesn't have any heart diseases, 1 if the patient has a heart disease

6) ever_married: "No" or "Yes"

7) work_type: "children", "Govt_jov", "Never_worked", "Private" or "Self-employed"

8) Residence_type: "Rural" or "Urban"

9) avg_glucose_level: average glucose level in blood

10) bmi: body mass index

11) smoking_status: "formerly smoked", "never smoked", "smokes" or "Unknown"*

12) stroke: 1 if the patient had a stroke or 0 if not

Pipeline adaptation

Now we follow the same algorithm in this dataset.

After splitting our dataset into training data and test data we see that cpu takes 5min to predict the classes to test samples.

Our algorithm shows 95.49% accuracy for this kind of dataset which means this algorithm identifies approx 96 % data correctly which is a good and f1 score is 93.