Big Home Work – Lazy FCA

Trofimov Nikolay

1.1 Choose DataSets

Let's look at the following datasets in order to apply the lazy fca algorithm and compare it with other machine learning algorithms

1) Go To College Dataset
link: https://www.kaggle.com/datasets/saddamazyazy/go-to-college-dataset
target: go to college or not

2) Car Insurance Claim Prediction
link : https://www.kaggle.com/datasets/ifteshanajnin/carinsuranceclaimpredictionclassification
target: variable indicating whether the policyholder _les a claim in the next 6 months or not.

3) Water Quality
link: https://www.kaggle.com/datasets/adityakadiwal/water-potability
target : water is safe or not

1.2 Feature Selecting

Let us list the main methods used to select features. For example, we will demonstrate on the first dataset.

The first method is building a correlation matrix for numerical features. As seen in Figure 1, we have the correlation matrix. Now we can exclude highly correlated features from each other.

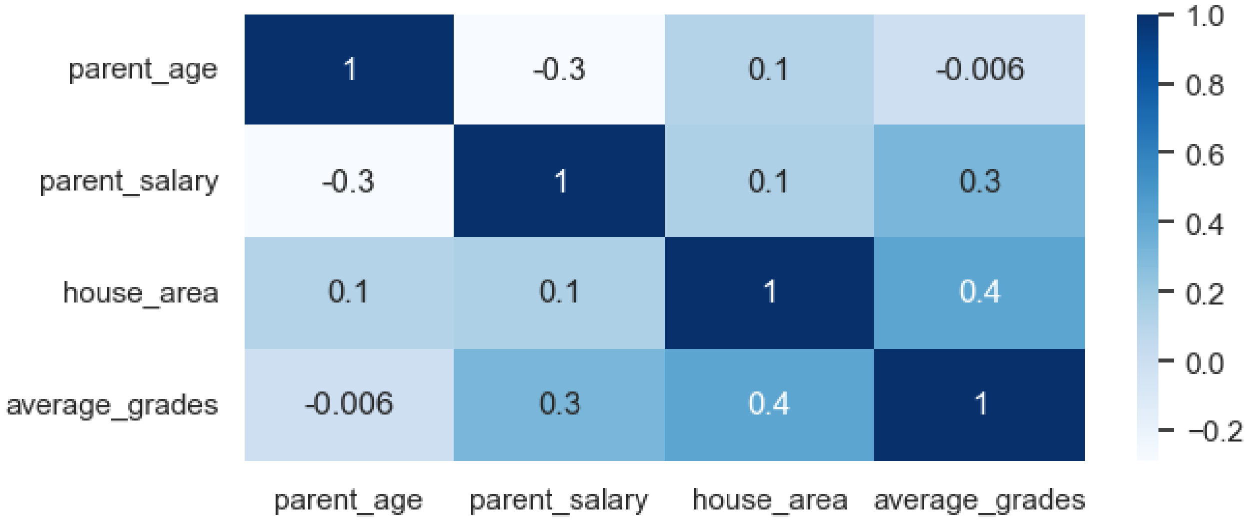


Figure 1. Correlation Matrix

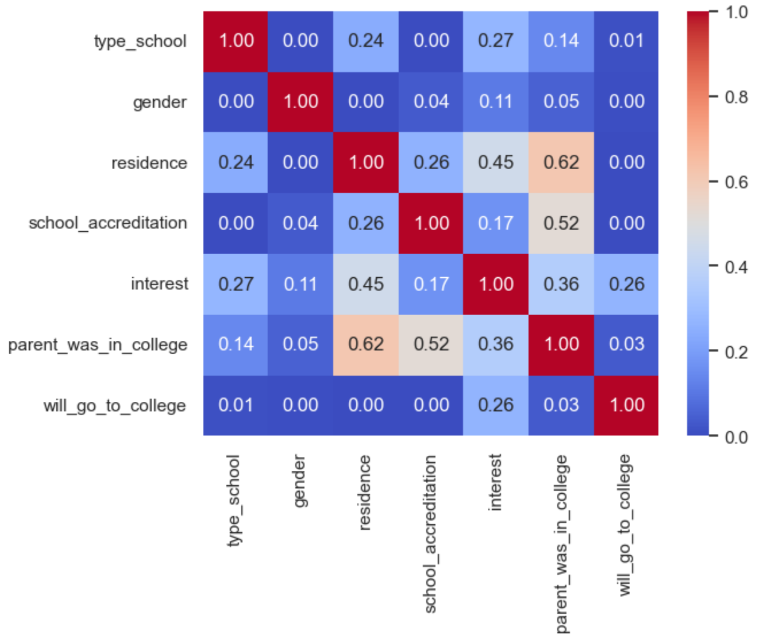


Figure 2. Cramer's V correlation matrix

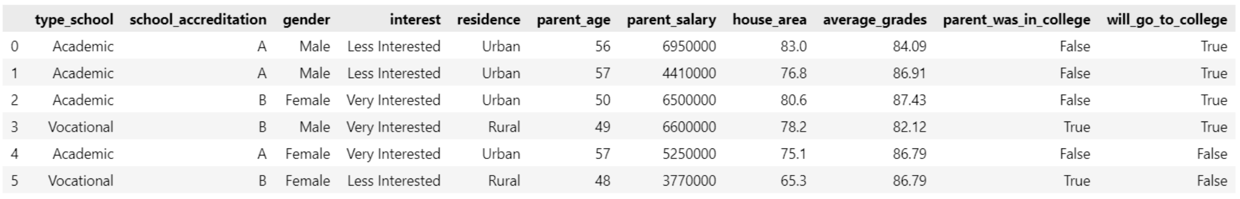
The second method is constructing a Cramer's V correlation matrix for the categorical features. Afterwards, as shown in Figure 2, we are able to identify the best categorical feature to learn from.

1.3 Preprocessing data

Let's give an example of data processing on the first dataset. We will use two techniques: One hot encoding and factorization.

We need to convert attributes such as ‘type_school’, ‘gender’, ‘residence’, ‘parent_was_in_college’ into numerical data. To do this, use the one hot encoding algorithm. Other categorical features such as ‘interest’ and ‘school_accreditation’ need to be encoded with different numbers in order to preserve the order between some values.

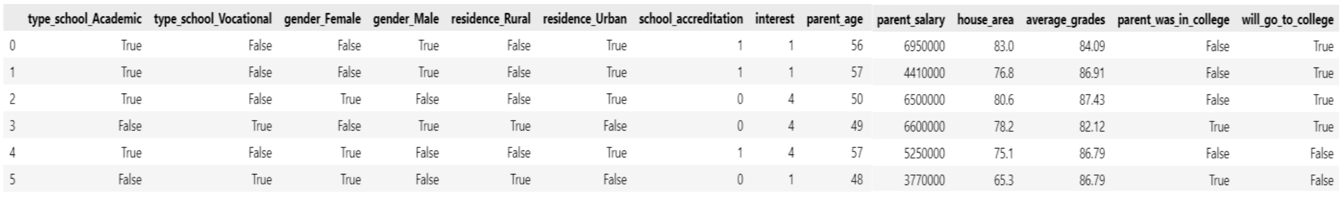
A data table with information about the selected students can be seen in Figure 3



	type_school	school_accreditation	gender	interest	residence	parent_age	parent_salary	house_area	average_grades	parent_was_in_college	will_go_to_college
0	Academic	A	Male	Less Interested	Urban	56	6950000	83.0	84.09	False	True
1	Academic	A	Male	Less Interested	Urban	57	4410000	76.8	86.91	False	True
2	Academic	B	Female	Very Interested	Urban	50	6500000	80.6	87.43	False	True
3	Vocational	B	Male	Very Interested	Rural	49	6600000	78.2	82.12	True	True
4	Academic	A	Female	Very Interested	Urban	57	5250000	75.1	86.79	False	False
5	Vocational	B	Female	Less Interested	Rural	48	3770000	65.3	86.79	True	False

Figure 3. Original data table

Before learning from data, it is necessary to perform a transformation on categorical features. We need to convert attributes such as ‘type_school’, ‘gender’, ‘residence’, ‘parent_was_in_college’ into numerical data. To do this, use the one hot encoding algorithm. Other categorical features such as ‘interest’ and ‘school_accreditation’ need to be encoded with different numbers in order to preserve the order between some values. The preprocessed data table can be seen in Figure 4

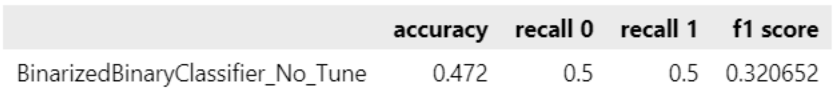


	type_school_Academic	type_school_Vocational	gender_Female	gender_Male	residence_Rural	residence_Urban	school_accreditation	interest	parent_age	parent_salary	house_area	average_grades	parent_was_in_college	will_go_to_college
0	True	False	False	True	False	True	1	1	56	6950000	83.0	84.09	False	True
1	True	False	False	True	False	True	1	1	57	4410000	76.8	86.91	False	True
2	True	False	True	False	False	True	0	4	50	6500000	80.6	87.43	False	True
3	False	True	False	True	True	False	0	4	49	6600000	78.2	82.12	True	True
4	True	False	True	False	False	True	1	4	57	5250000	75.1	86.79	False	False
5	False	True	True	False	True	False	0	1	48	3770000	65.3	86.79	True	False

Figure 4. Preprocessed data table

2.1 Using BinarizedBinaryClassifier and PatternBinaryClassifier

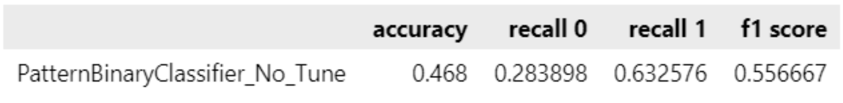
After preprocessing the data, we can use the BinarizedBinaryClassifier algorithm for categorical features. Also, before starting, we will select the alpha = 0.2 parameter, after receiving the result, we will calculate the main metrics as shown in Figure 5



	accuracy	recall 0	recall 1	f1 score
BinarizedBinaryClassifier_No_Tune	0.472	0.5	0.5	0.320652

Figure 5. Result of BinarizedBinaryClassifier algorithm, alpha = 0.2

Now let's choose the PatternBinaryClassifier algorithm with the alpha parameter also equal to 0.2. The results of the corresponding metrics are shown in Figure 6



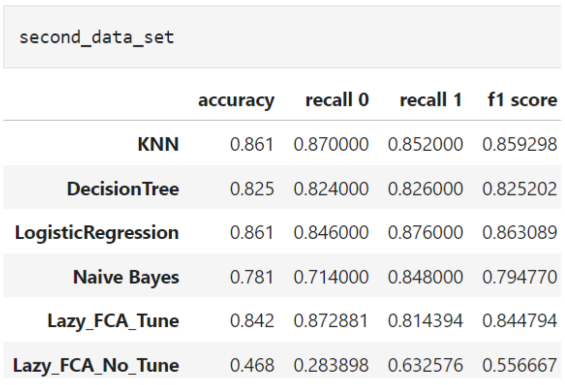
	accuracy	recall 0	recall 1	f1 score
PatternBinaryClassifier_No_Tune	0.468	0.283898	0.632576	0.556667

Figure 5. Result of PatternBinaryClassifier algorithm, alpha = 0.2

As we can see, the difference is not very big between the values of the metrics, only the F1-score has been increased. In the future, we will use the second algorithm as a priority, since it also works with numerical features.

2.2 Tune parameters and comparison with other algorithms

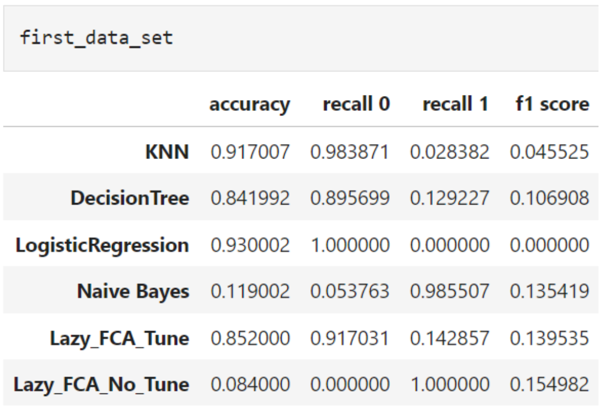
Definitely, the alpha parameter can indeed be selected using a greedy algorithm. To do so, we will need to write a special function which will iterate over alpha and select the best value based on the k-fold cross-validation. After implementing this function, it's important to verify that it's working correctly. Additionally, as a result of this verification, it's essential to compare it to other machine learning algorithms, such as K-nearest neighbors, decision trees, naive Bayes, and logistic regression. This comparison will help us determine whether or not it's profitable for us to utilize our algorithm. Let's look at the results for all datasets in Figures 6-8



```
second_data_set
```

	accuracy	recall 0	recall 1	f1 score
KNN	0.861	0.870000	0.852000	0.859298
DecisionTree	0.825	0.824000	0.826000	0.825202
LogisticRegression	0.861	0.846000	0.876000	0.863089
Naive Bayes	0.781	0.714000	0.848000	0.794770
Lazy_FCA_Tune	0.842	0.872881	0.814394	0.844794
Lazy_FCA_No_Tune	0.468	0.283898	0.632576	0.556667

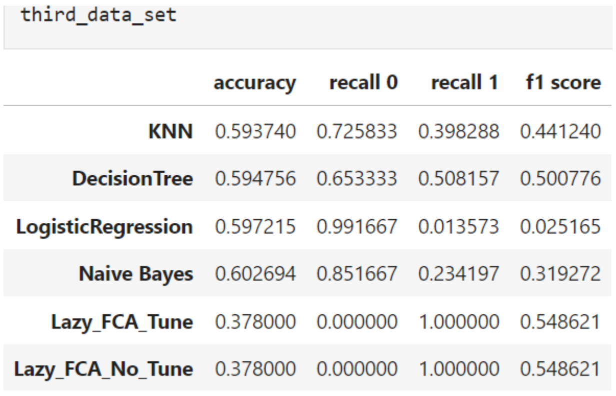
Figure 6. Result for the first Dataset



```
first_data_set
```

	accuracy	recall 0	recall 1	f1 score
KNN	0.917007	0.983871	0.028382	0.045525
DecisionTree	0.841992	0.895699	0.129227	0.106908
LogisticRegression	0.930002	1.000000	0.000000	0.000000
Naive Bayes	0.119002	0.053763	0.985507	0.135419
Lazy_FCA_Tune	0.852000	0.917031	0.142857	0.139535
Lazy_FCA_No_Tune	0.084000	0.000000	1.000000	0.154982

Figure 7. Result for the second Dataset



```
third_data_set
```

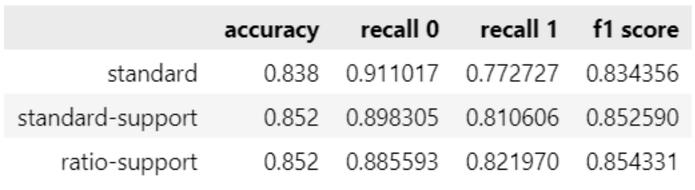
	accuracy	recall 0	recall 1	f1 score
KNN	0.593740	0.725833	0.398288	0.441240
DecisionTree	0.594756	0.653333	0.508157	0.500776
LogisticRegression	0.597215	0.991667	0.013573	0.025165
Naive Bayes	0.602694	0.851667	0.234197	0.319272
Lazy_FCA_Tune	0.378000	0.000000	1.000000	0.548621
Lazy_FCA_No_Tune	0.378000	0.000000	1.000000	0.548621

Figure 8. Result for the third Dataset

In every case, we increased the accuracy metric. As you can see in our tuned model, the accuracy metric is the greatest.

2.3 Different methods of algorithm

Let's look at the metric values of different decisions functions in our lazy algorithm. For this, we use the first dataset to calculate predictions using "standard", "standard-support", and "ratio-support". We then display the results in a table, as shown in Figure 9.



	accuracy	recall 0	recall 1	f1 score
standard	0.838	0.911017	0.772727	0.834356
standard-support	0.852	0.898305	0.810606	0.852590
ratio-support	0.852	0.885593	0.821970	0.854331

Figure 9. Different decision functions

As you can see, the results are approximately the same, but the selection functions "standard-support" and "ratio-support" still showed a better performance than "standard".

3.1 Conclusion

Based on this comparison, it can be concluded that the Lazy FCA algorithm produces results similar to other ML algorithms. Also, its ability to select a decision function improves the model.