

Ordered Sets for Data Analysis Project Report

December 2023
Moscow, Russia

Author: Marwan Bouabidi



NATIONAL RESEARCH
UNIVERSITY

Department of Computer Science

Contents

1	Introduction	3
2	Models	3
2.1	Datasets	3
2.2	Classification	4
2.2.1	Standard ML models	4
2.2.2	Neural FCA model	5
3	Results	6
4	Conclusion	8
5	Appendix	9
5.1	Logs	9

List of Figures

1	Few objects from each dataset	4
2	Example of a binarised numerical target attribute (salary in USD, testing data)	6
3	Fitted edges NN for the 3rd binary target attribute (salary in usd) in dataset 1 during the 4th KFold	8

List of Tables

1	Average accuracy scores	6
2	Average F1 scores	6
3	Accuracy scores for Neural FCA	7
4	F1 scores for Neural FCA	7
5	Average scores for Neural FCA	7

1 Introduction

Neural Networks (NNs) are well known Machine Learning models, arguably the most popular, used in almost all fields nowadays. This is due to their excellent performances at analysing large sets of data from all kinds of systems, and providing fairly accurate forecasts of chosen target variables. But Despite this success, their performances remain hard to interpret most of the times. That is why a Formal Concept Analysis (FCA) approach is introduced to respond to the need of interpretability in NNs, and AI in general.

FCA, being a mathematical approach for analyzing structures and sets, brings order to complex datasets by identifying meaningful relationships, therefore shedding some light on the hidden complex patterns in data, while the Neural Network offers its reliable predictive power. By integrating the two together, the aim is to offer predictions that are not only reliable but also rational, which means there is some form of reasoning behind them.

In this project, we'll be exploring an FCA and neural network hybrid model (Neural FCA), and comparing its performance to some standard ML algorithms.

2 Models

2.1 Datasets

The following datasets were used:

- <https://www.kaggle.com/datasets/inductiveanks/employee-salaries-for-different-job-roles>
- <https://www.kaggle.com/datasets/mexwell/us-energy-production>
- <https://www.kaggle.com/datasets/iamsouravbanerjee/world-population-dataset>

Objective: testing the Neural FCA method's performance to classifying data.

```

Unnamed: 0  work_year experience_level employment_type ... employee_residence remote_ratio company_location company_size
0          0          2020                MI          FT ...                DE          0          DE          L
1          1          2020                SE          FT ...                JP          0          JP          S
2          2          2020                SE          FT ...                GB          50          GB          M
3          3          2020                MI          FT ...                HN          0          HN          S
4          4          2020                SE          FT ...                US          50          US          L

[5 rows x 12 columns]

Treating ds_salaries data...
Rank CCA3 Country/Territory Capital ... Area (km²) Density (per km²) Growth Rate World Population Percentage
0    36  AFG  Afghanistan      Kabul ...  652230      63.0587      1.0257      0.52
1   138  ALB  Albania          Tirana ...  28748      98.8702      0.9957      0.04
2    34  DZA  Algeria          Algiers ... 2381741      18.8531      1.0164      0.56
3   213  ASM  American Samoa  Pago Pago ...    199      222.4774      0.9831      0.00
4   203  AND  Andorra          Andorra la Vella ...    468      170.5641      1.0100      0.00

[5 rows x 17 columns]

Treating world_population data...
Utility.Number Utility.Name Utility.State ... Retail.Total.Revenue Retail.Total.Sales Retail.Total.Customers
0          34  City of Abbeville - (SC)      SC ...      7536.0      58000.0      3844.0
1          55  City of Aberdeen - (MS)      MS ...     14797.0     204261.0     3229.0
2          59  City of Abbeville - (LA)      LA ...     12383.0     127579.0     5494.0
3          84  A & N Electric Coop          VA ...     78507.0     704010.0    35934.0
4          87  City of Ada - (MN)           MN ...      1593.0      20028.0     1185.0

```

Figure 1: Few objects from each dataset

2.2 Classification

2.2.1 Standard ML models

The following standard ML classification techniques were applied to the previous data:

- Decision Trees
- Random Forests
- kNN
- XGBoost

The chosen attributes (and the target attributes) can be seen in the code below:

```

datasets = ['ds_salaries', 'world_population', 'US_electricity_2017']
ds_salaries_features = ['job_title', 'experience_level', 'salary_in_usd',
                        'work_year', 'employment_type', 'remote_ratio', 'company_size']
world_population_features = ['Country/Territory', 'Rank', 'Continent',
                             'Growth Rate', '2022 Population', '2020 Population', '2015 Population',
                             '2010 Population', '2000 Population', 'Area (km2)', 'Density (per km2)']
US_electricity_features = ['Utility.Number', 'Utility.Type',
                           'Demand.Summer Peak', 'Sources.Total', 'Uses.Total',
                           'Retail.Total.Customers']
targets = [ds_salaries_features[2], world_population_features[3],
           US_electricity_features[2]]

```

Before classification, the data had to be binarised. For categorical data in string format, it was achieved with this code:

```
dataPath = r'datasets/{}.csv'.format(dataset_name)
df = pd.read_csv(dataPath)
processed_df = pd.DataFrame()
for feature in features:
    #Detect if there are strings in the feature's column
    non_numeric_columns = df.select_dtypes(exclude=['number']).columns.tolist()
    if(feature in non_numeric_columns):
        #Check if binarization is possible, by checking category number
        binarized_df_column = pd.get_dummies(df[feature], prefix=feature)
        bin_categories = binarized_df_column.columns.tolist()
        if(len(bin_categories) <= 20):
            #Add to processed_df
            processed_df = pd.concat([processed_df, binarized_df_column], axis=1)
    else:
        processed_df = pd.concat([processed_df, df[feature]], axis=1)
```

As we can see, the categorical attributes with more than 20 categories have been filtered out, since they will raise the execution time significantly (attribute country in dataset 2 for example. But in that dataset, every country have a unique rank associated with it, so the information is not lost) And the target attribute (numerical in the 3 datasets), had to be binarised as well, into $N = 6$ different intervals:

```
def NumOneHotEncoder(df_target, n, target):
    min = df_target.min()
    max = df_target.max()
    step = abs(max-min)/n
    bin_edges = [min+i*step for i in range(n+1)]
    bin_labels = [r'{}_{}'.format(target, i) for i in range(n)]
    categorical_df_target = pd.cut(df_target, bins=bin_edges,
                                   labels=bin_labels)
    return pd.get_dummies(categorical_df_target)
```

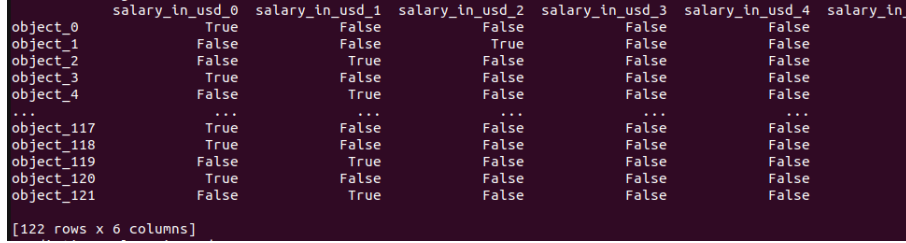
2.2.2 Neural FCA model

The Neural FCA method relies on formal concept analysis for enhanced interpretability and understanding of the inner layers of the neural network.

Again, the same target attributes have been chosen, and the data has been divided into training and testing sets, but for the Neural FCA algorithm in this case, only binary features are allowed, so all attributes have been binarised.

The target attributes are numerical in the 3 datasets, therefore requiring to be binarised with the **interval scaling** method. This produces a target attribute with multiple binary columns (one for each interval, the number of intervals is the variable N mentioned previously), but the neural FCA algorithm only

accepts a single binary column as target. So the solution was to predict all columns of the binarised target, one by one, and then assembling them together into multiple columns to match the format of the binarised (with interval scaling) target attribute.



```

salary_in_usd_0 salary_in_usd_1 salary_in_usd_2 salary_in_usd_3 salary_in_usd_4 salary_in_
object_0         True         False         False         False         False         salary_in_
object_1         False        False         True          False         False         False
object_2         False        True          False         False         False         False
object_3         True         False        False         False         False         False
object_4         False        True          False         False         False         False
...             ...          ...          ...          ...          ...
object_117       True         False        False         False         False         False
object_118       True         False        False         False         False         False
object_119       False        True          False         False         False         False
object_120       True         False        False         False         False         False
object_121       False        True          False         False         False         False
[122 rows x 6 columns]

```

Figure 2: Example of a binarised numerical target attribute (salary in USD, testing data)

Using the algorithm **Sofia** specifically to build the concept lattice reduced the execution time significantly compared to the default algorithm.

3 Results

With 5 KFold partitions, and 6 category intervals for the target attributes ($N = 6$), these results were computed for the standard ML methods:

ML method	DS Salaries	World Population	US electricity 2017
Decision Trees	0.539	0.761	0.932
Random Forests	0.581	0.773	0.934
kNN	0.573	0.718	0.917
XGBoost	0.586	0.743	0.933

Table 1: Average accuracy scores

ML method	DS Salaries	World Population	US electricity 2017
Decision Trees	0.586	0.764	0.955
Random Forests	0.599	0.787	0.957
kNN	0.595	0.722	0.95
XGBoost	0.612	0.784	0.957

Table 2: Average F1 scores

As for the Neural FCA model, also using 5 KFold partitions and 6 category

intervals for the target attributes, I got:

Datasets	KFold1	KFold2	KFold3	KFold4	KFold5
DS Salaries	0	0.008	0.43	0.446	0.397
World Population	0.745	0.766	0.66	0.702	0.674
US electricity	0.98	0.986	0.976	0.965	0.688

Table 3: Accuracy scores for Neural FCA

Datasets	KFold1	KFold2	KFold3	KFold4	KFold5
DS Salaries	0	0	0.43	0.446	0.397
World Population	0.745	0.766	0.66	0.702	0.681
US electricity	0.983	0.986	0.979	0.974	0.812

Table 4: F1 scores for Neural FCA

Dataset	Average Accuracy	Average F1 score
DS Salaries	0.256	0.254
World Population	0.709	0.711
US electricity	0.918	0.947

Table 5: Average scores for Neural FCA

We can notice that the results are slightly worse than those given by the standard ML models for all the used datasets (I executed all the models under the same conditions), and it was noticeably worse for the first dataset. Also, despite using the algorithm 'Sofia' which helped a lot in terms of execution time, the neural FCA algorithm was still much slower, mainly due to the fact that it's unable to take more than one binary target attribute at a time, and I had to execute it multiple times for every numerical attribute, each for one of its many binarised vectors.

Naturally, by reducing the discretisation of the target attribute to $N = 3$ for dataset 1 (to better visualise the neural network), the accuracy and f1 score improved significantly. But the results we get are less useful, as 3 intervals seem to be not enough information about a salary:

Accuracy scores:

[0.9180327868852459, 0.9180327868852459, 0.9504132231404959, 0.9008264462809917, 0.8677685950413223]

Average: 0.9110147676466603

F1 scores:

[0.9180327868852459, 0.9218106995884774, 0.9504132231404959, 0.9008264462809917,

0.8677685950413223]
Average: 0.9117703501873067

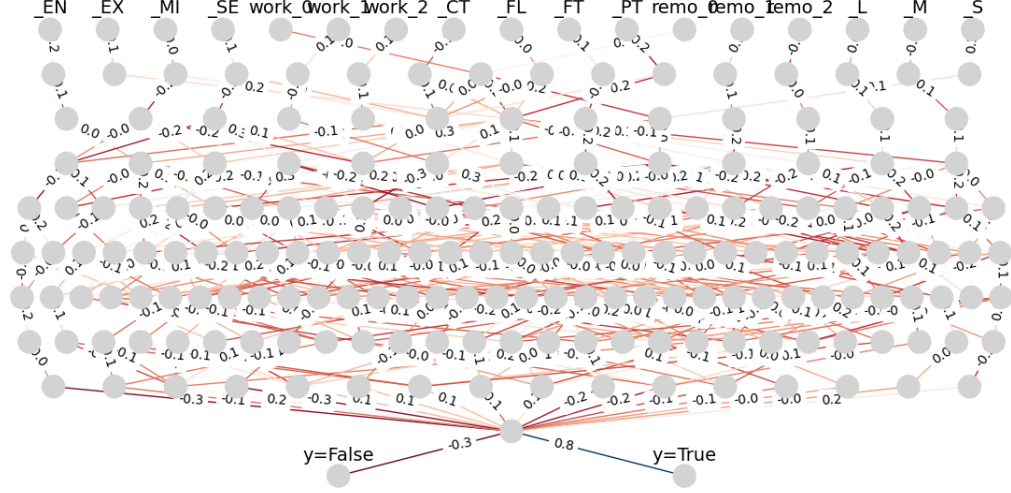


Figure 3: Fitted edges NN for the 3rd binary target attribute (salary in usd) in dataset 1 during the 4th KFold

On the other hand, reducing the number of futures in any of the datasets resulted in lower accuracy and f1 scores.

4 Conclusion

By comparing the accuracy and F1 scores, we can see that the Neural FCA algorithm gives slightly worse results than the standard ML models used in the beginning, but it is still accurate, while also adding needed interpretability to the Neural Network behind its predictions.

The execution time heavily depends on the algorithm used to build the concept lattice as it was mentioned before. But, even despite using a good algorithm, the execution time was still slower than that of the other models.

The neural FCA model is unable to take more than one binary target attribute at a time, unlike the other models used in this project. So, to predict a binarised numerical target, it has to be executed as many times as there are binary columns for this target.

While reducing the number of discretisation bins of the target attribute raises accuracy and f1 scores, the target variable is of less quality overall.

5 Appendix

The full code is available on my GitHub repository: <https://github.com/MaruwanDono/NeuralFCA-Classification.git>

5.1 Logs

Execution logs for $N = 6$:

Treating ds_salaries data...

Neural FCA classification for:

	experience_level_EN	experience_level_EX	...	company_size_M	company_size_S
0	0	0	...	0	0
1	0	0	...	0	1
2	0	0	...	1	0
3	0	0	...	0	1
4	0	0	...	0	0

[5 rows x 14 columns]

Neural FCA algorithm: KFold 1...

Predicting salary_in_usd_0...

Predicting salary_in_usd_1...

Predicting salary_in_usd_2...

Predicting salary_in_usd_3...

Predicting salary_in_usd_4...

Predicting salary_in_usd_5...

Neural FCA algorithm: KFold 2...

Predicting salary_in_usd_0...

Predicting salary_in_usd_1...

Predicting salary_in_usd_2...

Predicting salary_in_usd_3...

Predicting salary_in_usd_4...

Predicting salary_in_usd_5...

Neural FCA algorithm: KFold 3...

Predicting salary_in_usd_0...

Predicting salary_in_usd_1...

Predicting salary_in_usd_2...

Predicting salary_in_usd_3...

Predicting salary_in_usd_4...

Predicting salary_in_usd_5...

Neural FCA algorithm: KFold 4...

Predicting salary_in_usd_0...

Predicting salary_in_usd_1...

Predicting salary_in_usd_2...

Predicting salary_in_usd_3...

Predicting salary_in_usd_4...

```

Predicting salary_in_usd_5...
Neural FCA algorithm: KFold 5...
Predicting salary_in_usd_0...
Predicting salary_in_usd_1...
Predicting salary_in_usd_2...
Predicting salary_in_usd_3...
Predicting salary_in_usd_4...
Predicting salary_in_usd_5...
Accuracy scores:
[0.0, 0.00819672131147541, 0.4297520661157025, 0.4462809917355372,
0.39669421487603307]
Average: 0.25618479880774964
F1 scores:
[0.0, 0.0, 0.4297520661157025, 0.4462809917355372,
0.39669421487603307]
Average: 0.2545454545454545

Treating world_population data...
Neural FCA classification for:

```

	Rank	Continent_Africa	...	Area (km2)	Density (per km2)
0	36	0	...	652230	63.0587
1	138	0	...	28748	98.8702
2	34	1	...	2381741	18.8531
3	213	0	...	199	222.4774
4	203	0	...	468	170.5641

```

[5 rows x 15 columns]
Neural FCA algorithm: KFold 1...
Predicting Growth Rate_0...
Predicting Growth Rate_1...
Predicting Growth Rate_2...
Predicting Growth Rate_3...
Predicting Growth Rate_4...
Predicting Growth Rate_5...
Neural FCA algorithm: KFold 2...
Predicting Growth Rate_0...
Predicting Growth Rate_1...
Predicting Growth Rate_2...
Predicting Growth Rate_3...
Predicting Growth Rate_4...
Predicting Growth Rate_5...
Neural FCA algorithm: KFold 3...
Predicting Growth Rate_0...
Predicting Growth Rate_1...
Predicting Growth Rate_2...
Predicting Growth Rate_3...

```

```

Predicting Growth Rate_4...
Predicting Growth Rate_5...
Neural FCA algorithm: KFold 4...
Predicting Growth Rate_0...
Predicting Growth Rate_1...
Predicting Growth Rate_2...
Predicting Growth Rate_3...
Predicting Growth Rate_4...
Predicting Growth Rate_5...
Neural FCA algorithm: KFold 5...
Predicting Growth Rate_0...
Predicting Growth Rate_1...
Predicting Growth Rate_2...
Predicting Growth Rate_3...
Predicting Growth Rate_4...
Predicting Growth Rate_5...
Accuracy scores:
[0.7446808510638298, 0.7659574468085106, 0.6595744680851063, 0.7021276595744681,
0.6739130434782609]
Average: 0.7092506938020351
F1 scores:
[0.7446808510638298, 0.7659574468085105, 0.6595744680851063, 0.7021276595744681,
0.6813186813186812]
Average: 0.7107318213701193

```

Treating US_electricity_2017 data...

Neural FCA classification for:

	Utility.Number	...	Retail.Total.Customers
0	34	...	3844.0
1	55	...	3229.0
2	59	...	5494.0
3	84	...	35934.0
4	87	...	1185.0

[5 rows x 14 columns]

```

Neural FCA algorithm: KFold 1...
Predicting Demand.Summer Peak_0...
Predicting Demand.Summer Peak_1...
Predicting Demand.Summer Peak_2...
Predicting Demand.Summer Peak_3...
Predicting Demand.Summer Peak_4...
Predicting Demand.Summer Peak_5...
Neural FCA algorithm: KFold 2...
Predicting Demand.Summer Peak_0...
Predicting Demand.Summer Peak_1...
Predicting Demand.Summer Peak_2...

```

```

Predicting Demand.Summer Peak_3...
Predicting Demand.Summer Peak_4...
Predicting Demand.Summer Peak_5...
Neural FCA algorithm: KFold 3...
Predicting Demand.Summer Peak_0...
Predicting Demand.Summer Peak_1...
Predicting Demand.Summer Peak_2...
Predicting Demand.Summer Peak_3...
Predicting Demand.Summer Peak_4...
Predicting Demand.Summer Peak_5...
Neural FCA algorithm: KFold 4...
Predicting Demand.Summer Peak_0...
Predicting Demand.Summer Peak_1...
Predicting Demand.Summer Peak_2...
Predicting Demand.Summer Peak_3...
Predicting Demand.Summer Peak_4...
Predicting Demand.Summer Peak_5...
Neural FCA algorithm: KFold 5...
Predicting Demand.Summer Peak_0...
Predicting Demand.Summer Peak_1...
Predicting Demand.Summer Peak_2...
Predicting Demand.Summer Peak_3...
Predicting Demand.Summer Peak_4...
Predicting Demand.Summer Peak_5...
Accuracy scores:
[0.9795275590551181, 0.9811023622047244, 0.9763779527559056, 0.9653543307086614,
0.6876971608832808]
Average: 0.918011873121538
F1 scores:
[0.9826224328593995, 0.985759493670886, 0.9786898184688241, 0.9737887212073073,
0.8119180633147114]
Average: 0.9465557059042256

```