# OSDA Big Homework: Neural FCA

Marwan Bouabidi, M1 DS

Dec 2023

## 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.

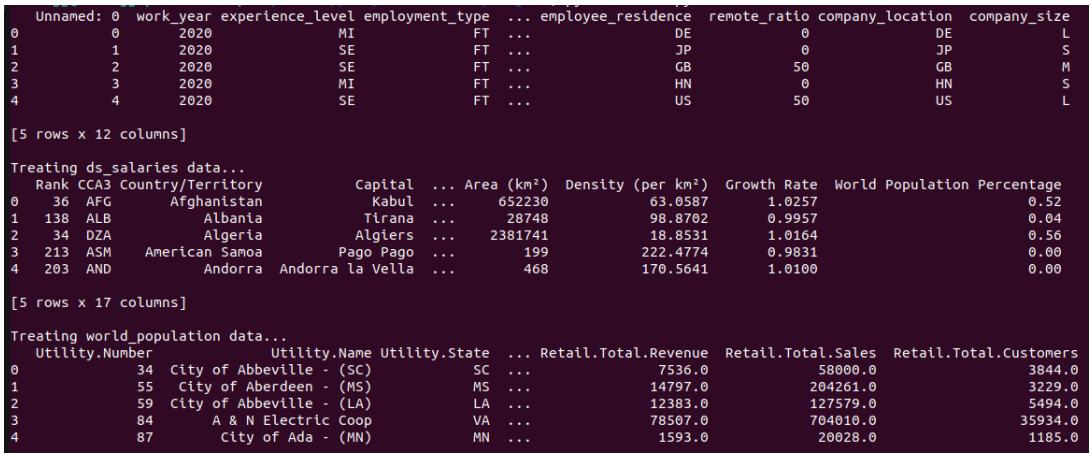
## 2 Models

### 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',
```

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

```
non_numeric_columns = df.select_dtypes(exclude=['number']).columns.tolist()
            if(feature in non_numeric_columns):
                binarized_df_column = pd.get_dummies(df[feature])
```

And the target attribute (numerical in the 3 datasets), had to be binarised 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 | |
| object_1 | False | False | True | False | False | |
| object_2 | False | True | False | False | False | |
| object_3 | True | False | False | False | False | |
| object_4 | False | True | False | False | False | |
| ... | ... | ... | ... | ... | ... | |
| object_117 | True | False | False | False | False | |
| object_118 | True | False | False | False | False | |
| object_119 | False | True | False | False | False | |
| object_120 | True | False | False | False | False | |
| object_121 | False | True | False | False | False | |

[122 rows x 6 columns]

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

# 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

# 4 Conclusion

By comparing the accuracy and F1 scores, we can see that the Neural FCA algorithm gives similar results to the standard ML models employed in the beginning, with similar execution time, all while adding needed interpretability to the Neural Network behind its predictions.
You can find the full code on my GitHub repository here.