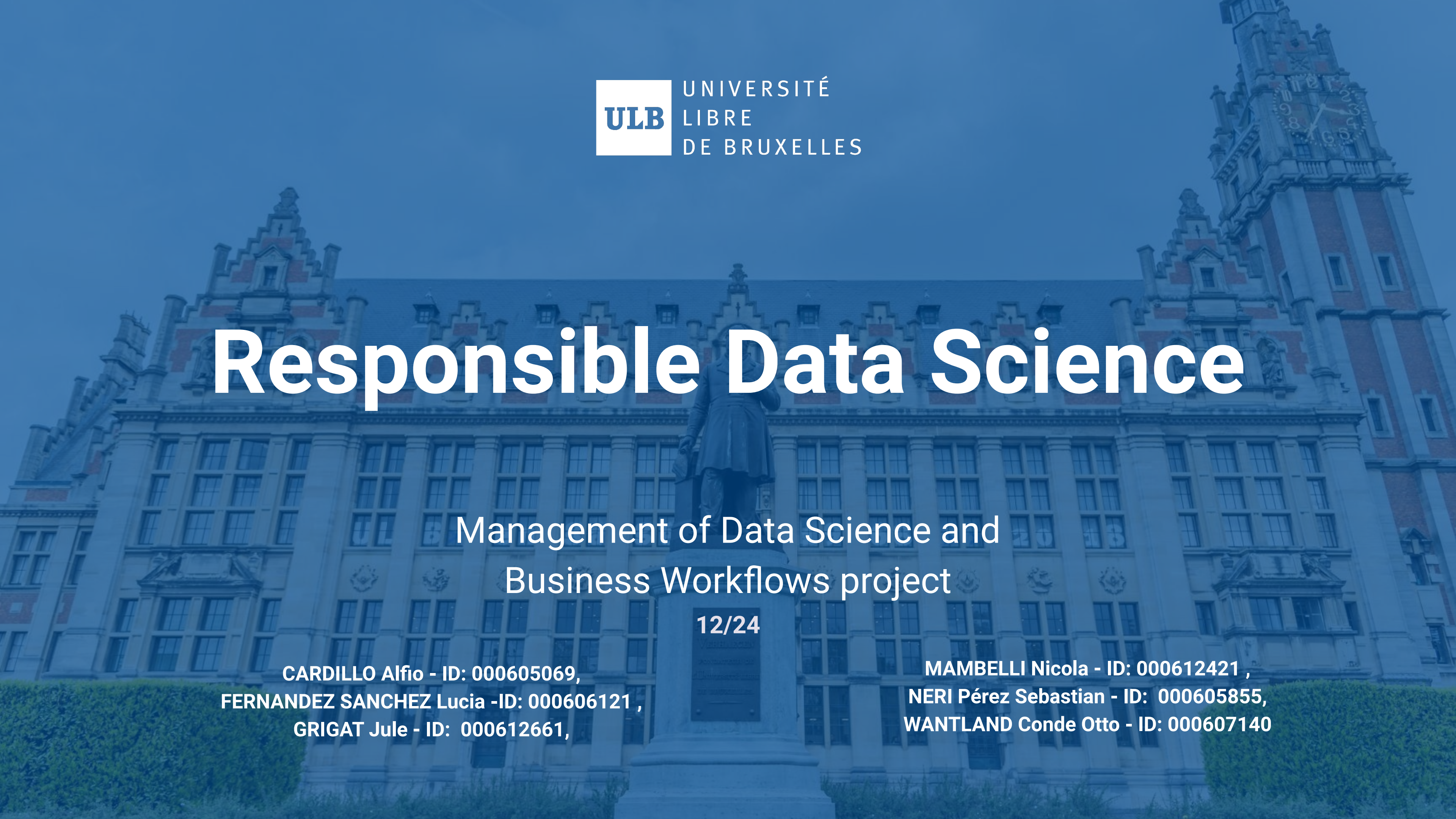


UNIVERSITÉ LIBRE DE BRUXELLES

Responsible Data Science

Management of Data Science and Business Workflows project

12/24

CARDILLO Alfio - ID: 000605069,
FERNANDEZ SANCHEZ Lucia -ID: 000606121 ,
GRIGAT Jule - ID: 000612661,

MAMBELLI Nicola - ID: 000612421 ,
NERI Pérez Sebastian - ID: 000605855,
WANTLAND Conde Otto - ID: 000607140

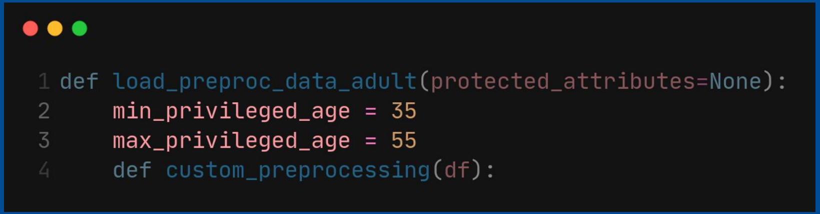
UNIVERSITÉ LIBRE DE BRUXELLES

1. Classification

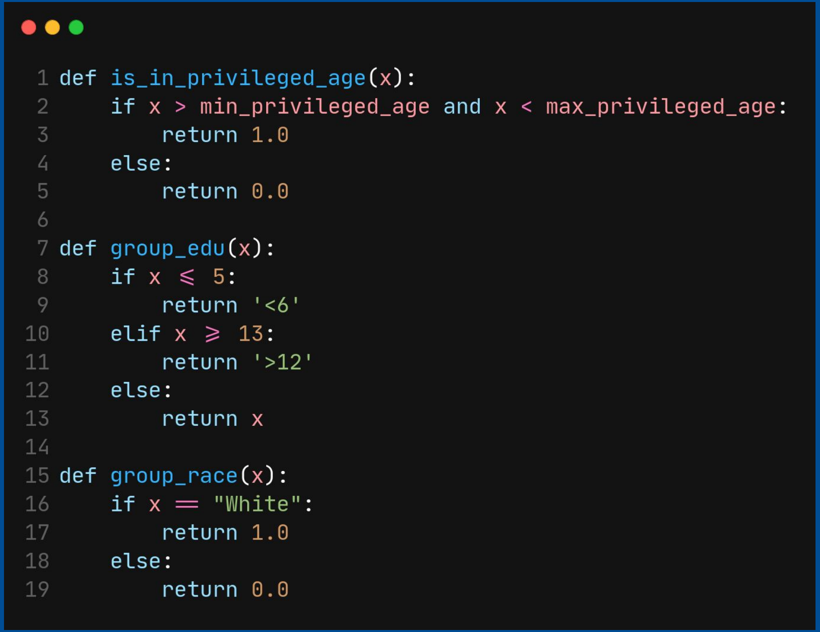
ULB UNIVERSITÉ LIBRE DE BRUXELLES

Preprocessing

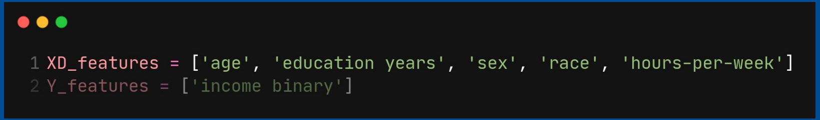
- AIF360 preprocessing template
- Binarize age
- Add hours-per-week



```python
def load_preproc_data_adult(protected_attributes=None):
    min_privileged_age = 35
    max_privileged_age = 55
    def custom_preprocessing(df):
```



```python
def is_in_privileged_age(x):
    if x > min_privileged_age and x < max_privileged_age:
        return 1.0
    else:
        return 0.0

def group_edu(x):
    if x <= 5:
        return '<6'
    elif x >= 13:
        return '>12'
    else:
        return x

def group_race(x):
    if x == "White":
        return 1.0
    else:
        return 0.0
```

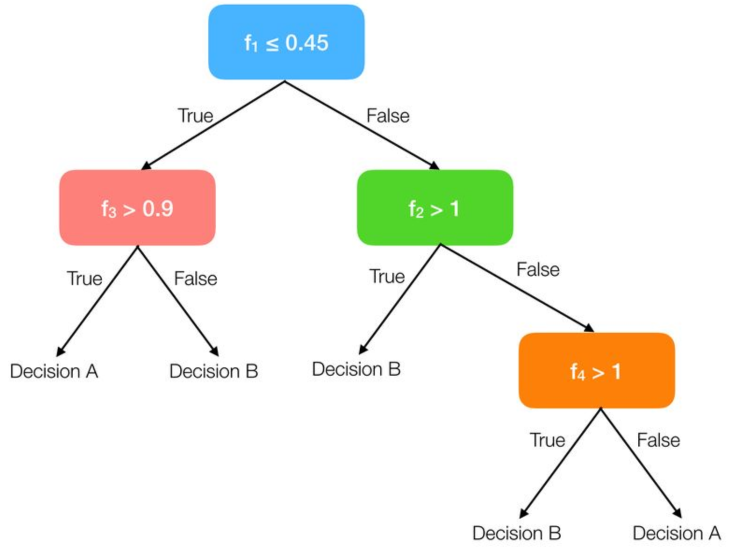


```python
XD_features = ['age', 'education years', 'sex', 'race', 'hours-per-week']
Y_features = ['income binary']
```

ULB UNIVERSITÉ LIBRE DE BRUXELLES

Decision Tree

- Classifier predicts probability, with the validation set we choose the best threshold.
- Balanced accuracy ~ 0.72
- Adding more features, increased the accuracy, but in some cases had some other impact.



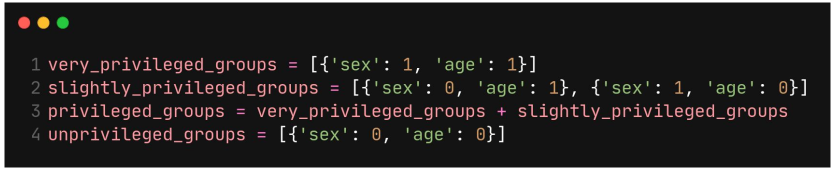
UNIVERSITÉ LIBRE DE BRUXELLES

2. Fairness

UNIVERSITÉ LIBRE DE BRUXELLES

Privileged definition

- Very privileged
- Slightly privileged
- Privileged
- Unprivileged



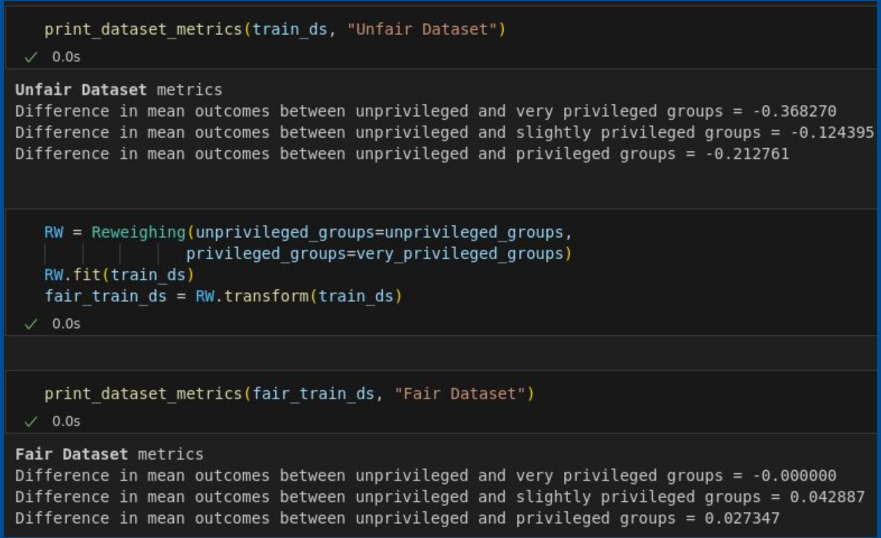
```
very_privileged_groups = [{'sex': 1, 'age': 1}]
slightly_privileged_groups = [{'sex': 0, 'age': 1}, {'sex': 1, 'age': 0}]
privileged_groups = very_privileged_groups + slightly_privileged_groups
unprivileged_groups = [{'sex': 0, 'age': 0}]
```

UNIVERSITÉ LIBRE DE BRUXELLES

Reweighing

We actually end up penalizing the slightly and very privileged groups

$$Pr(Y = 1|D = \text{unprivileged}) - Pr(Y = 1|D = \text{privileged})$$

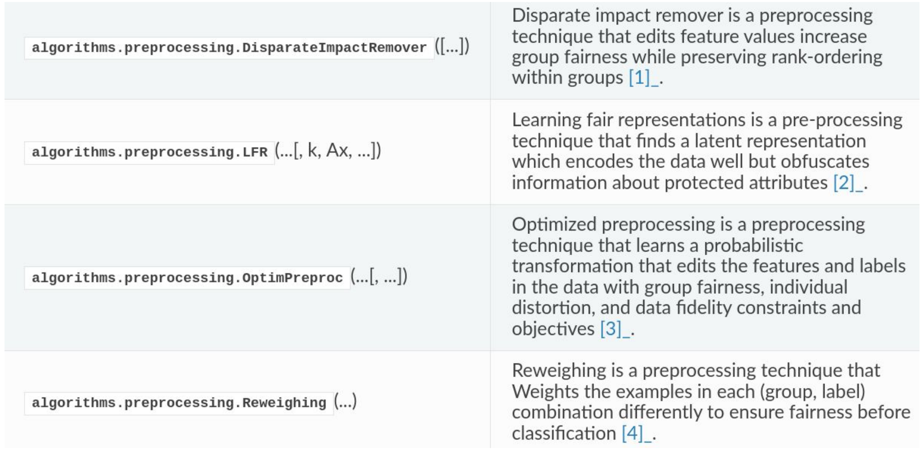


```
print_dataset_metrics(train_ds, "Unfair Dataset")
✓ 0.0s

Unfair Dataset metrics
Difference in mean outcomes between unprivileged and very privileged groups = -0.368270
Difference in mean outcomes between unprivileged and slightly privileged groups = -0.124395
Difference in mean outcomes between unprivileged and privileged groups = -0.212761

RW = Reweighing(unprivileged_groups=unprivileged_groups,
                privileged_groups=very_privileged_groups)
RW.fit(train_ds)
fair_train_ds = RW.transform(train_ds)
✓ 0.0s

print_dataset_metrics(fair_train_ds, "Fair Dataset")
✓ 0.0s

Fair Dataset metrics
Difference in mean outcomes between unprivileged and very privileged groups = -0.000000
Difference in mean outcomes between unprivileged and slightly privileged groups = 0.042887
Difference in mean outcomes between unprivileged and privileged groups = 0.027347
```

UNIVERSITÉ LIBRE DE BRUXELLES

Other fairness algorithm

- Disparate Impact Remover: edit features
- LFR: obfuscates info about protected attributes
- OptimPreproc: edit features



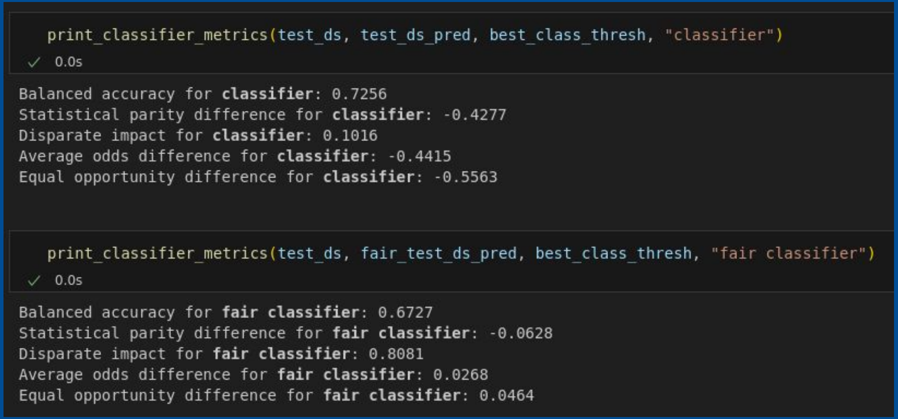
`algorithms.preprocessing.DisparateImpactRemover` ([...])	Disparate impact remover is a preprocessing technique that edits feature values increase group fairness while preserving rank-ordering within groups [1]_.
`algorithms.preprocessing.LFR` (...[, k, Ax, ...])	Learning fair representations is a pre-processing technique that finds a latent representation which encodes the data well but obfuscates information about protected attributes [2]_.
`algorithms.preprocessing.OptimPreproc` (...[, ...])	Optimized preprocessing is a preprocessing technique that learns a probabilistic transformation that edits the features and labels in the data with group fairness, individual distortion, and data fidelity constraints and objectives [3]_.
`algorithms.preprocessing.Reweighing` (...)	Reweighing is a preprocessing technique that Weights the examples in each (group, label) combination differently to ensure fairness before classification [4]_.

Fair classifier

- Lower accuracy
- Better fairness metrics

$$DI = \frac{Pr(Y = 1|D = \text{unprivileged})}{Pr(Y = 1|D = \text{privileged})}$$

$$EOR = TPR_{D=\text{unprivileged}} - TPR_{D=\text{privileged}}$$

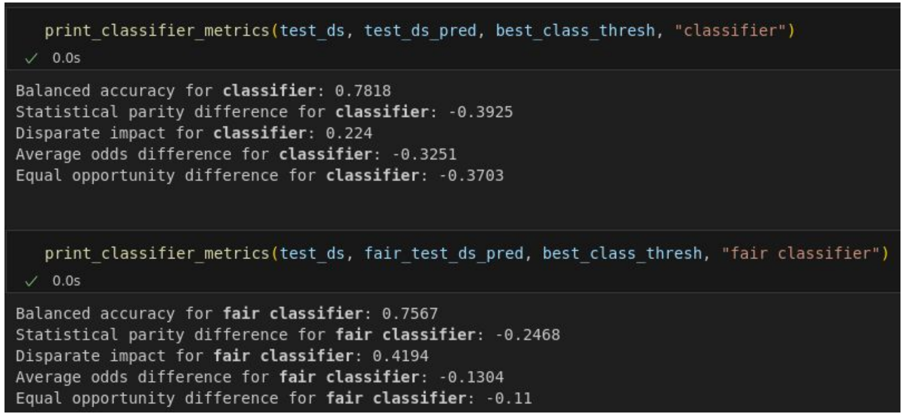


```
print_classifier_metrics(test_ds, test_ds_pred, best_class_thresh, "classifier")
✓ 0.0s

Balanced accuracy for classifier: 0.7256
Statistical parity difference for classifier: -0.4277
Disparate impact for classifier: 0.1016
Average odds difference for classifier: -0.4415
Equal opportunity difference for classifier: -0.5563
```

```
print_classifier_metrics(test_ds, fair_test_ds_pred, best_class_thresh, "fair classifier")
✓ 0.0s

Balanced accuracy for fair classifier: 0.6727
Statistical parity difference for fair classifier: -0.0628
Disparate impact for fair classifier: 0.8081
Average odds difference for fair classifier: 0.0268
Equal opportunity difference for fair classifier: 0.0464
```

What if we add marital status?

Possible connection with protected attribute age



```
print_classifier_metrics(test_ds, test_ds_pred, best_class_thresh, "classifier")
✓ 0.0s

Balanced accuracy for classifier: 0.7818
Statistical parity difference for classifier: -0.3925
Disparate impact for classifier: 0.224
Average odds difference for classifier: -0.3251
Equal opportunity difference for classifier: -0.3703
```

```
print_classifier_metrics(test_ds, fair_test_ds_pred, best_class_thresh, "fair classifier")
✓ 0.0s

Balanced accuracy for fair classifier: 0.7567
Statistical parity difference for fair classifier: -0.2468
Disparate impact for fair classifier: 0.4194
Average odds difference for fair classifier: -0.1304
Equal opportunity difference for fair classifier: -0.11
```

UNIVERSITÉ LIBRE DE BRUXELLES

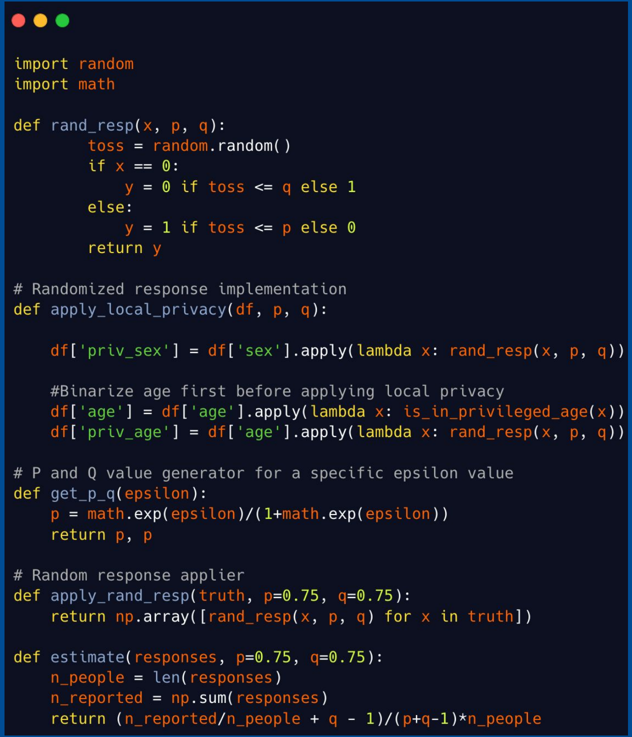
3. Privacy

UNIVERSITÉ LIBRE DE BRUXELLES

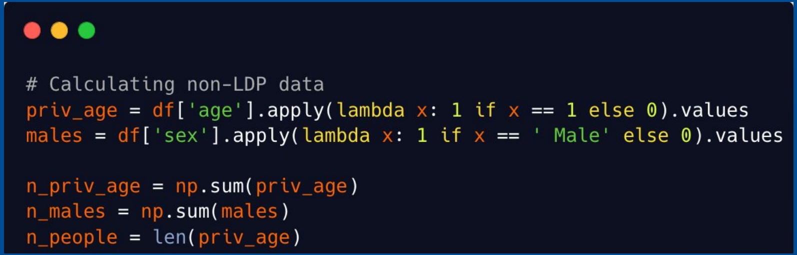
Implementing LDP

- Using Randomized Response

$$r = \frac{1}{p+q-1}\left(\frac{n_1^{rep}}{n} + q - 1\right).$$



```
import random
import math

def rand_resp(x, p, q):
        toss = random.random()
        if x == 0:
            y = 0 if toss <= q else 1
        else:
            y = 1 if toss <= p else 0
        return y

# Randomized response implementation
def apply_local_privacy(df, p, q):

    df['priv_sex'] = df['sex'].apply(lambda x: rand_resp(x, p, q))

    #Binarize age first before applying local privacy
    df['age'] = df['age'].apply(lambda x: is_in_privileged_age(x))
    df['priv_age'] = df['age'].apply(lambda x: rand_resp(x, p, q))

# P and Q value generator for a specific epsilon value
def get_p_q(epsilon):
    p = math.exp(epsilon)/(1+math.exp(epsilon))
    return p, p

# Random response applier
def apply_rand_resp(truth, p=0.75, q=0.75):
    return np.array([rand_resp(x, p, q) for x in truth])

def estimate(responses, p=0.75, q=0.75):
    n_people = len(responses)
    n_reported = np.sum(responses)
    return (n_reported/n_people + q - 1)/(p+q-1)*n_people
```

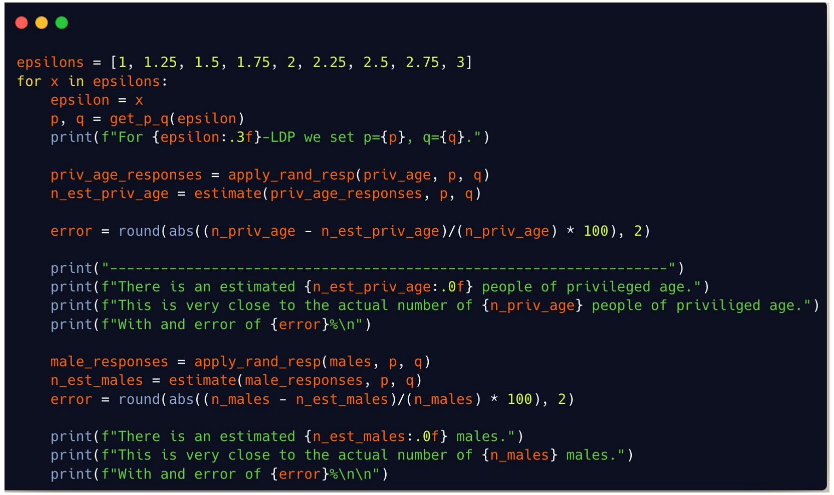


```
# Calculating non-LDP data
priv_age = df['age'].apply(lambda x: 1 if x == 1 else 0).values
males = df['sex'].apply(lambda x: 1 if x == ' Male' else 0).values

n_priv_age = np.sum(priv_age)
n_males = np.sum(males)
n_people = len(priv_age)
```

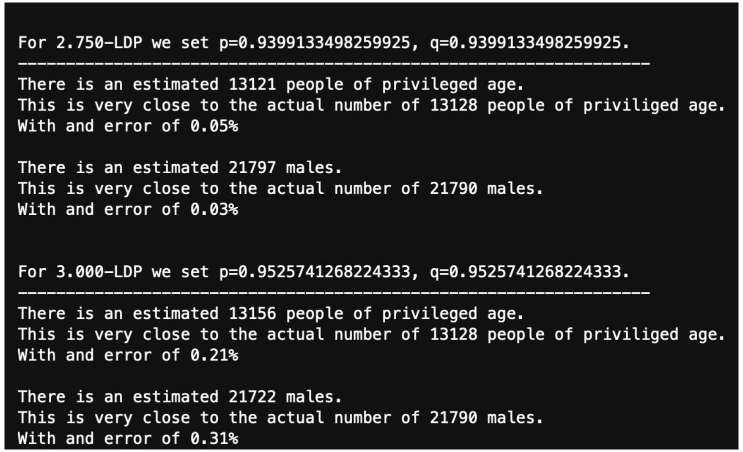
ULB UNIVERSITÉ LIBRE DE BRUXELLES

Results for Different Epsilon Values

Testing different values to find the best P and Q for our evaluation.



```
epsilons = [1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3]
for x in epsilons:
    epsilon = x
    p, q = get_p_q(epsilon)
    print(f"For {epsilon:.3f}-LDP we set p={p}, q={q}.")

    priv_age_responses = apply_rand_resp(priv_age, p, q)
    n_est_priv_age = estimate(priv_age_responses, p, q)

    error = round(abs((n_priv_age - n_est_priv_age)/(n_priv_age) * 100), 2)

    print("-------------------------------------------------------------------")
    print(f"There is an estimated {n_est_priv_age:.0f} people of privileged age.")
    print(f"This is very close to the actual number of {n_priv_age} people of priviliged age.")
    print(f"With and error of {error}%\n")

    male_responses = apply_rand_resp(males, p, q)
    n_est_males = estimate(male_responses, p, q)
    error = round(abs((n_males - n_est_males)/(n_males) * 100), 2)

    print(f"There is an estimated {n_est_males:.0f} males.")
    print(f"This is very close to the actual number of {n_males} males.")
    print(f"With and error of {error}%\n\n")
```

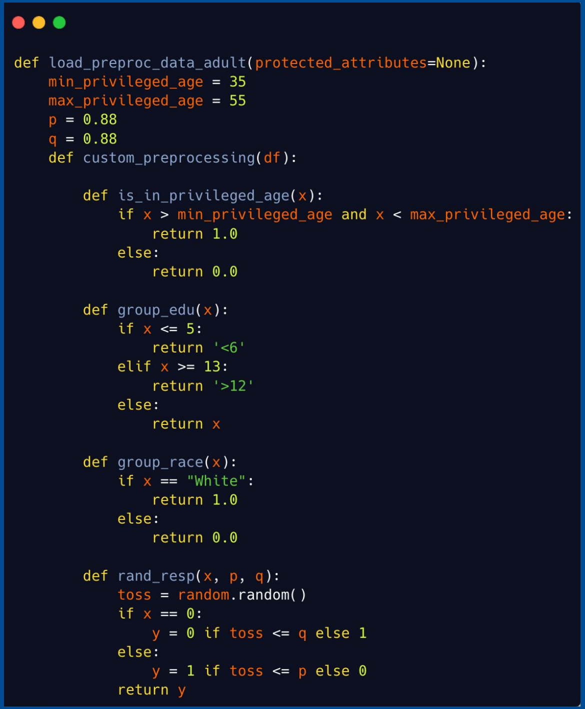


```
For 2.750-LDP we set p=0.9399133498259925, q=0.9399133498259925.
-------------------------------------------------------------------
There is an estimated 13121 people of privileged age.
This is very close to the actual number of 13128 people of priviliged age.
With and error of 0.05%

There is an estimated 21797 males.
This is very close to the actual number of 21790 males.
With and error of 0.03%


For 3.000-LDP we set p=0.9525741268224333, q=0.9525741268224333.
-------------------------------------------------------------------
There is an estimated 13156 people of privileged age.
This is very close to the actual number of 13128 people of priviliged age.
With and error of 0.21%

There is an estimated 21722 males.
This is very close to the actual number of 21790 males.
With and error of 0.31%
```

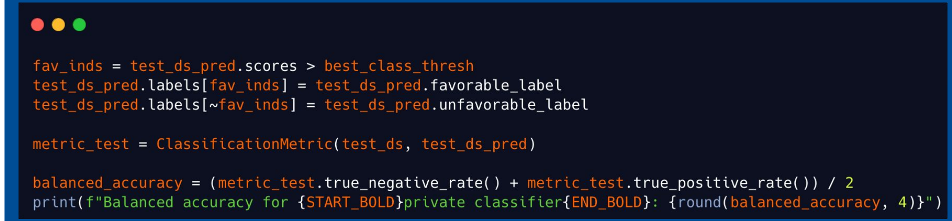
UNIVERSITÉ LIBRE DE BRUXELLES

Implementation Into Classifier

- Balanced accuracy for private classifier: **71%**
- Implementing LDP lowers classifier accuracy slightly.



```python
def load_preproc_data_adult(protected_attributes=None):
    min_privileged_age = 35
    max_privileged_age = 55
    p = 0.88
    q = 0.88
    def custom_preprocessing(df):

        def is_in_privileged_age(x):
            if x > min_privileged_age and x < max_privileged_age:
                return 1.0
            else:
                return 0.0

        def group_edu(x):
            if x <= 5:
                return '<6'
            elif x >= 13:
                return '>12'
            else:
                return x

        def group_race(x):
            if x == "White":
                return 1.0
            else:
                return 0.0

        def rand_resp(x, p, q):
            toss = random.random()
            if x == 0:
                y = 0 if toss <= q else 1
            else:
                y = 1 if toss <= p else 0
            return y
```

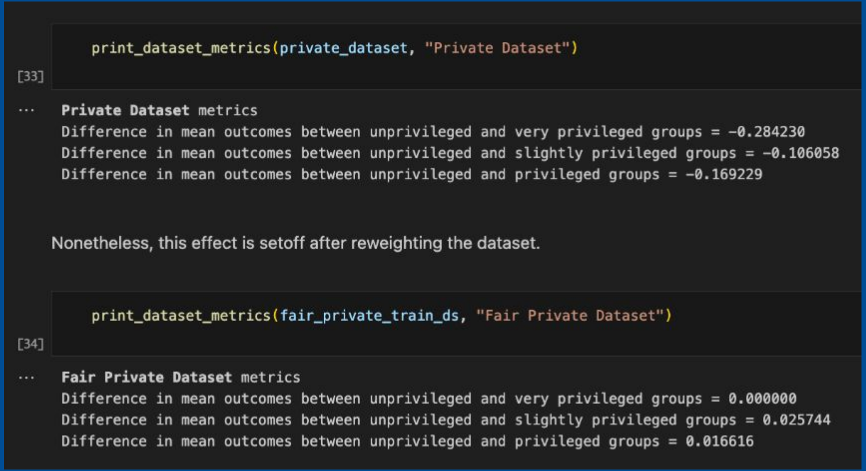


```python
fav_inds = test_ds_pred.scores > best_class_thresh
test_ds_pred.labels[fav_inds] = test_ds_pred.favorable_label
test_ds_pred.labels[~fav_inds] = test_ds_pred.unfavorable_label

metric_test = ClassificationMetric(test_ds, test_ds_pred)

balanced_accuracy = (metric_test.true_negative_rate() + metric_test.true_positive_rate()) / 2
print(f"Balanced accuracy for {START_BOLD}private classifier{END_BOLD}: {round(balanced_accuracy, 4)}")
```

4. Privacy + Fairness

Re-weighing of the Private Dataset

- Largest gap (-0.284) in private dataset: unprivileged vs. very privileged.
- Moderate gap (-0.169) in private dataset: unprivileged vs. privileged.
- Fair dataset: no gap (0.000) for unprivileged vs. very privileged.
- Small gap (0.026) in fair dataset: unprivileged vs. slightly privileged.
- Fair dataset reduces disparities significantly.



```
[33]  print_dataset_metrics(private_dataset, "Private Dataset")
```

```
Private Dataset metrics
Difference in mean outcomes between unprivileged and very privileged groups = -0.284230
Difference in mean outcomes between unprivileged and slightly privileged groups = -0.106058
Difference in mean outcomes between unprivileged and privileged groups = -0.169229
```

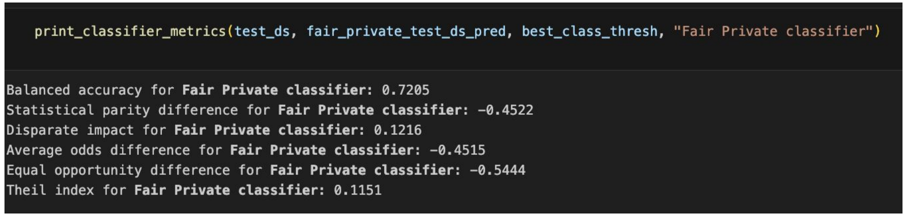
Nonetheless, this effect is setoff after reweighting the dataset.

```
[34]  print_dataset_metrics(fair_private_train_ds, "Fair Private Dataset")
```

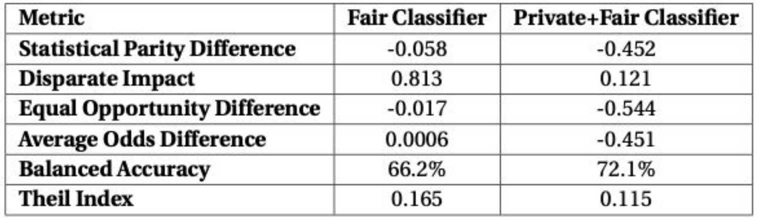
```
Fair Private Dataset metrics
Difference in mean outcomes between unprivileged and very privileged groups = 0.000000
Difference in mean outcomes between unprivileged and slightly privileged groups = 0.025744
Difference in mean outcomes between unprivileged and privileged groups = 0.016616
```

Results

- Private+Fair favors privileged groups more than Fair Classifier.
- Larger biases in Equal Opportunity show unequal treatment.
- Private+Fair has higher balanced accuracy (72.1%).
- Improved Theil Index (0.115) indicates better info distribution.
- Trade-off: fairness sacrificed for performance, raising ethical concerns.



```
print_classifier_metrics(test_ds, fair_private_test_ds_pred, best_class_thresh, "Fair Private classifier")

Balanced accuracy for Fair Private classifier: 0.7205
Statistical parity difference for Fair Private classifier: -0.4522
Disparate impact for Fair Private classifier: 0.1216
Average odds difference for Fair Private classifier: -0.4515
Equal opportunity difference for Fair Private classifier: -0.5444
Theil index for Fair Private classifier: 0.1151
```



Metric	Fair Classifier	Private+Fair Classifier
Statistical Parity Difference	-0.058	-0.452
Disparate Impact	0.813	0.121
Equal Opportunity Difference	-0.017	-0.544
Average Odds Difference	0.0006	-0.451
Balanced Accuracy	66.2%	72.1%
Theil Index	0.165	0.115

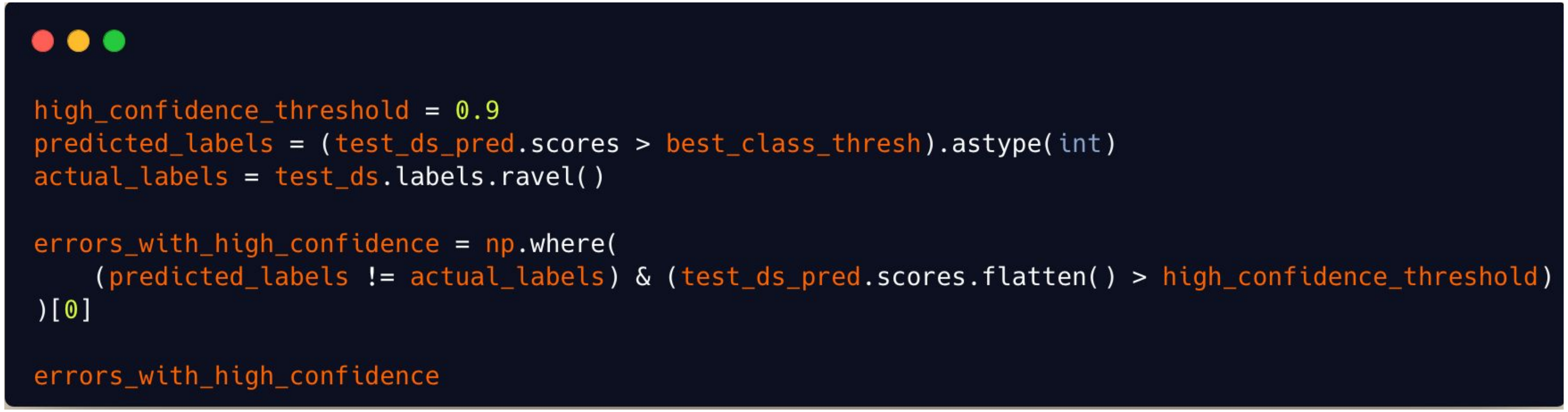
UNIVERSITÉ LIBRE DE BRUXELLES

5.Explainability

UNIVERSITÉ
LIBRE
DE BRUXELLES

Methodology

Study the explainability of the private classifier.



```
high_confidence_threshold = 0.9
predicted_labels = (test_ds_pred.scores > best_class_thresh).astype(int)
actual_labels = test_ds.labels.ravel()

errors_with_high_confidence = np.where(
    (predicted_labels != actual_labels) & (test_ds_pred.scores.flatten() > high_confidence_threshold)
)[0]

errors_with_high_confidence
```

array([0, 0, 0, ..., 7326, 7326, 7326])

UNIVERSITÉ LIBRE DE BRUXELLES

1. **Errors concentrated in certain characteristics:**

- This suggests that the errors occur primarily in specific subgroups, such as young males of nonprivileged age.

2. High but incorrect confidence

- Instances show predictions with high confidence (>= 0.6).

- The model might be overestimating its confidence in certain scenarios where key features, such as age and sex, are noisy or not well represented.

Recommendation

- The noise added for privacy may be disproportionately affecting certain groups, causing systematic errors in these categories.
- It is recommended to evaluate noise reduction in age and sex or adjust hyper parameters to improve the balance between privacy and accuracy.

UNIVERSITÉ LIBRE DE BRUXELLES

6. Explainability and LLM

Using the Explainability Method LIME

LIME is an explainability Method which generates feature-importance pairs showing how each attribute contributes to a prediction.

Steps taken for this part:

1. Choose an explainability method - LIME
2. Construct and configure the Model
3. Execute the model to explain the “errors with high confidence” from the previous part of the project
4. Print the results to a .json file
5. Let an LLM explain the results - Llama 3.2

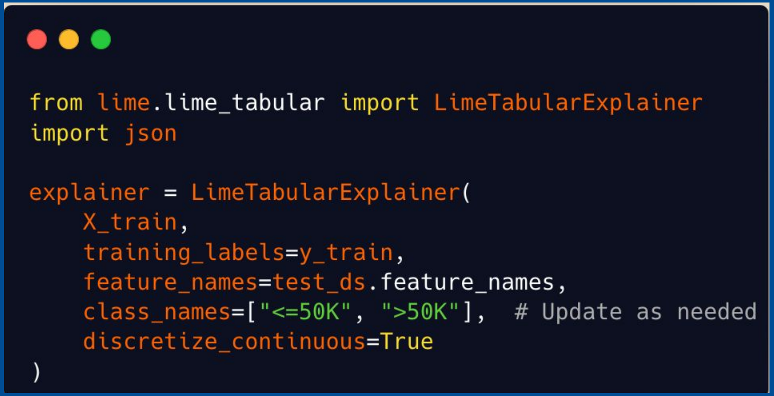
UNIVERSITÉ LIBRE DE BRUXELLES

Constructing the LIME Model

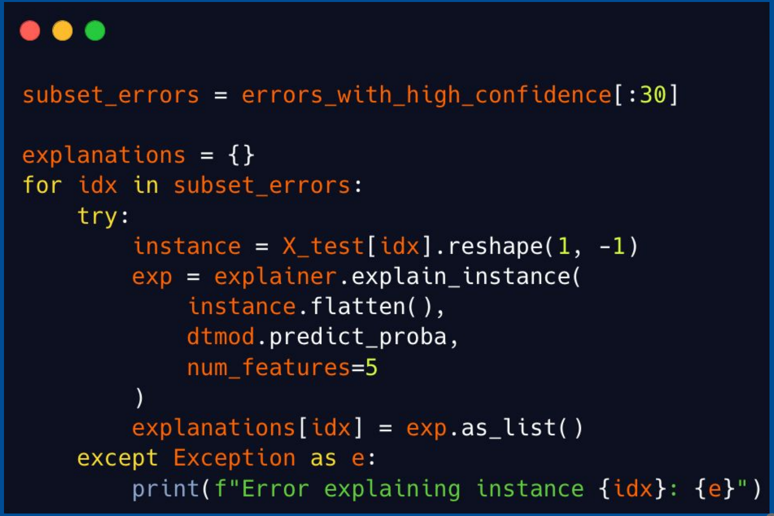
- Set Train data and Train Labels, and include attributes
- Set the classification attribute to <=50K and >50K

Running the Model

- Using the model to explore the subset of errors with high confidence.



```python
from lime.lime_tabular import LimeTabularExplainer
import json

explainer = LimeTabularExplainer(
    X_train,
    training_labels=y_train,
    feature_names=test_ds.feature_names,
    class_names=["<=50K", ">50K"],  # Update as needed
    discretize_continuous=True
)
```

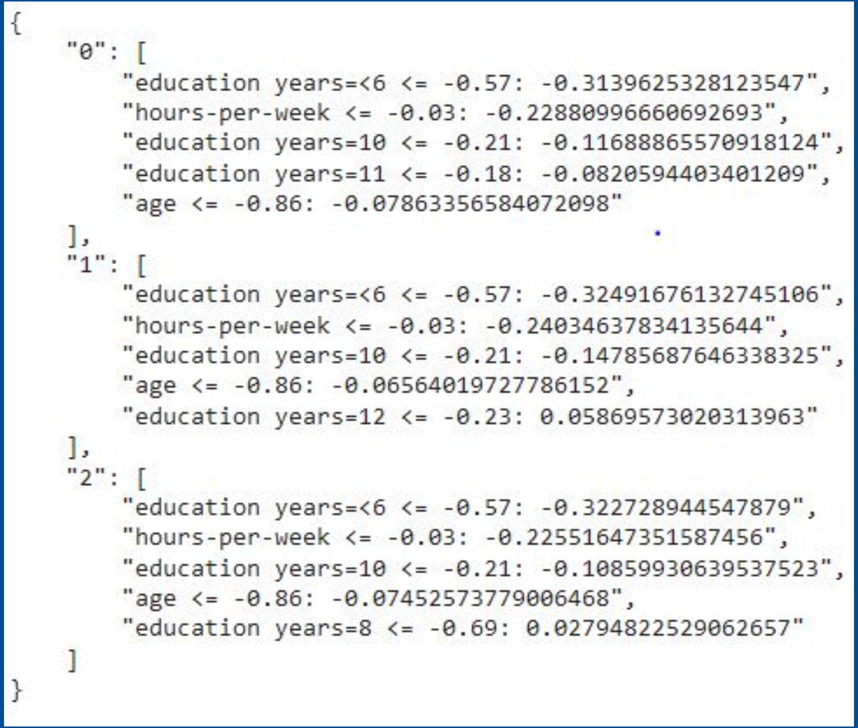


```python
subset_errors = errors_with_high_confidence[:30]

explanations = {}
for idx in subset_errors:
    try:
        instance = X_test[idx].reshape(1, -1)
        exp = explainer.explain_instance(
            instance.flatten(),
            dtmod.predict_proba,
            num_features=5
        )
        explanations[idx] = exp.as_list()
    except Exception as e:
        print(f"Error explaining instance {idx}: {e}")
```

Exemplary Model Output

- To be translated into understandable text by the LLM concluding:

  “The minimum education requirements for various jobs across different age groups and weekly working hours”



```
{
    "0": [
        "education years=<6 <= -0.57: -0.3139625328123547",
        "hours-per-week <= -0.03: -0.2288099660692693",
        "education years=10 <= -0.21: -0.1168886570918124",
        "education years=11 <= -0.18: -0.0820594403401209",
        "age <= -0.86: -0.0786335658407209"
    ],
    "1": [
        "education years=<6 <= -0.57: -0.3249167613274510",
        "hours-per-week <= -0.03: -0.2403463783413564",
        "education years=10 <= -0.21: -0.1478568764633832",
        "age <= -0.86: -0.0656401972778615",
        "education years=12 <= -0.23: 0.0586957302031396"
    ],
    "2": [
        "education years=<6 <= -0.57: -0.322728944547879",
        "hours-per-week <= -0.03: -0.2255164735158745",
        "education years=10 <= -0.21: -0.1085993063953752",
        "age <= -0.86: -0.0745257377900646",
        "education years=8 <= -0.69: 0.0279482252906265"
    ]
}
```

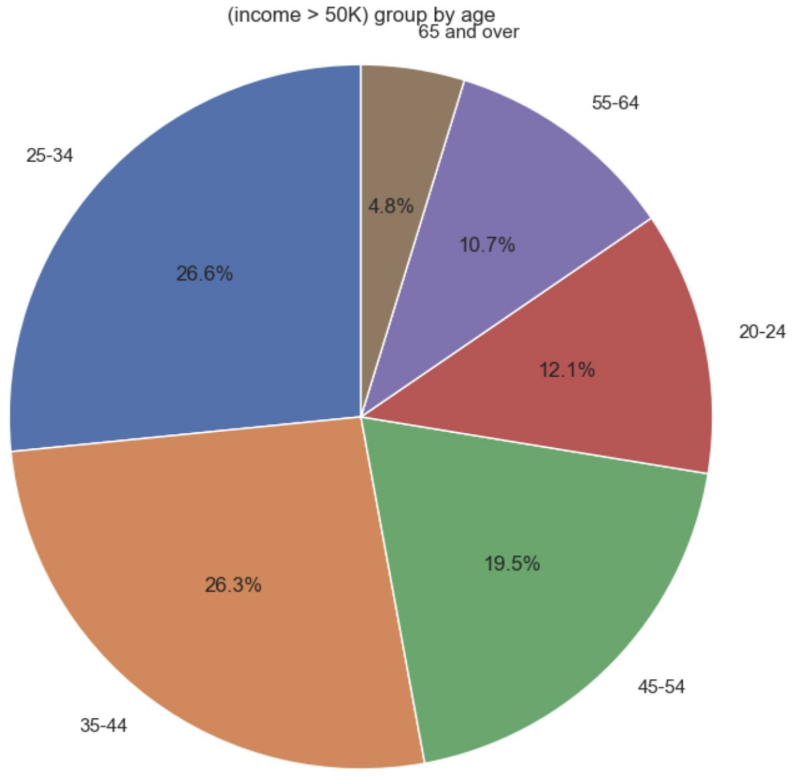
UNIVERSITÉ LIBRE DE BRUXELLES

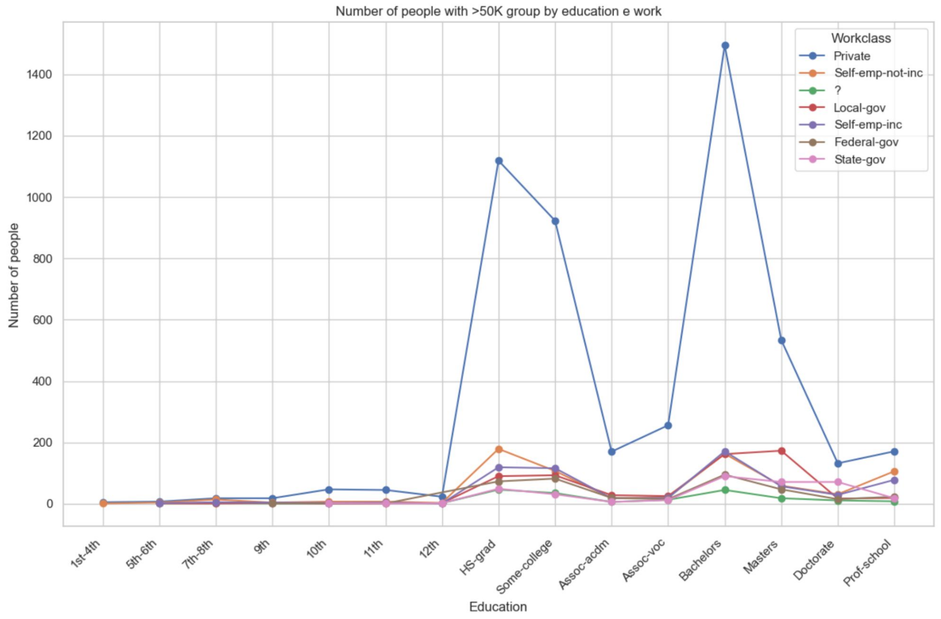
7. Free Exploration

UNIVERSITÉ LIBRE DE BRUXELLES

Income by age

The age group that represents the largest slice of the pie chart is the one that corresponds to people aged 25 to 34, making up 26.6% of the whole group





Income by Education and Work

This leads to the intuitive observation that, currently, for all jobs, the majority of people prefer to obtain a bachelor's degree.

Thank you for your attention!

Learn more at:
https://github.com/Carda01/responsibleDS_adult