



ECOLE
POLYTECHNIQUE
DE BRUXELLES

Erasmus Mundus joint Master in Big Data Management and Analytics

INFO-H420 Management of Data Science and Business Workflows

Responsible Data Science

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)

Prof. Sacharidis Dimitris

2024

1	Introduction	1
2	Classification	2
2.1	Overview	2
2.2	Pre-processing	2
2.3	Model	3
2.4	Result	3
3	Fairness	4
3.1	Overview	4
3.2	Privileged groups	4
3.3	Fair Classifier	4
3.4	Result	6
4	Privacy	8
4.1	Overview	8
4.2	Initial Exploration	8
4.3	Applying Local Differential Privacy	9
4.4	Classifier Integration	11
5	Privacy and Fairness	16
5.1	Overview	16
5.2	Methodology	16
5.3	Results	17
5.4	Recommendations	17
6	Explainability	19
6.1	Overview	19

6.2	Methodology	19
6.3	Results	20
7	Explainability and LLMs	22
7.1	Overview	22
7.2	Methodology	22
7.3	Results	24
8	Free Exploration	27
8.1	Overview	27
8.2	Income by age	27
8.3	Income by education and work	27
8.4	Income by gender	28
8.5	Income by gender and education	29
8.6	Income by Education year	29
9	Conclusion	30
	References	31

List of Tables

5.1 Comparison of Fairness and Performance Metrics between Fair Classifier and Private+Fair Classifier	17
--	----

List of Figures

2.1	The original dataframe	2
3.1	Privileged groups	5
3.2	Before and after reweighing	5
3.3	Fair classifier vs classifier	6
3.4	Classifiers with marital status	7
4.1	Cross-tabulation of initial values	8
4.2	Results of Epsilon Value Analysis	12
4.3	Cross-tabulation With LDP Applied	12
7.1	LLM-Configurations	25
8.1	Pie chart by Income and Age	27
8.2	Graph of Income group by education and work	28
8.3	Graph of income by gender	28
8.4	Graph of income by gender and education	29
8.5	Graph of income by education years	29

List of Codes

4.1	Randomized Response Implementation	9
4.2	Estimate Function	10
4.3	Estimate Function	10
4.4	Epsilon Value Comparison	11
4.5	Classifier With LDP	13
6.1	Threshold Definition	19
6.2	Predicted labels Definition	19
6.3	Actual labels Definition	19
6.4	Errors with high confidence	20
7.1	Install LIME as chosen explainability method	22
7.2	Actual labels Definition	23
7.3	Scan through top 10 errors with high confidence	23
7.4	Convert LIME explanations to a serializable format Definition	23
7.5	Output .JSON file	24

Nowadays, one of the biggest challenges that modern society faces is to find the optimal way to work on the big amount of Data that we have available.

Starting from getting the correct source to get the appropriate Data, trying to firstly get the information that the Data provides, as well as understand the meanings behind, then apply strategies where we manage to get more insights until we get to the place where we can predict the behavior of the new input data. Additionally, living in an era of Technology where bias, unfairness and in some cases hate, are being present in the decisions made by state-of-the-art Companies, or sometimes, by countries. For those reasons and more, studying a pipeline, its characteristics and implementation of the models for transforming Data is a crucial knowledge that every single person that daily works with Data, needs to understand.

Starting by classifying the input data, then continue by applying fairness, where we transform the classifier data to a fairer one. Continue by adding Privacy local differential privacy (LDP), where the input gets some transformation until the important data, the most private and risky one, gets changed. This method is very used in big companies such as Google, Apple and Microsoft [Wan18].

Explaining and creating a model with Natural Language processing, we will get hands-on experience in the virtues of a good performance at the same time as applying privacy in the crucial information. Last but not least, we are going to apply free will regarding the Dataset and find any new and interesting insights.

Overall, the importance of this project is to use a methodology for current the handy Data that is currently on hand, but if we are not critical of the performance, as well as notice the benefits of applying one transformation, we not only miss new opportunities to tackle, but also new ways of understanding the context that we already live on in the big Data world.

In the following Github link, you can find more details about the Code:

https://github.com/Carda01/responsibleDS_adult

2.1 Overview

Preprocess the data, and binarize Age. Split the data into train, validation, test sets, and train a classifier; we will refer to it as the classifier. Measure the performance of the **classifier** on the test set.

2.2 Pre-processing

The original data set contained 32561 rows each with 15 columns. Examples of records can be seen from 2.1 where we have displayed the first 10 rows of the adult data set.

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	income-per-year
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
5	37	Private	284582	Masters	14	Married-civ-spouse	Exec-managerial	Wife	White	Female	0	0	40	United-States	<=50K
6	49	Private	160187	9th	5	Married-spouse-absent	Other-service	Not-in-family	Black	Female	0	0	16	Jamaica	<=50K
7	52	Self-emp-not-inc	209642	HS-grad	9	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	45	United-States	>50K
8	31	Private	45781	Masters	14	Never-married	Prof-specialty	Not-in-family	White	Female	14084	0	50	United-States	>50K
9	42	Private	159449	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	5178	0	40	United-States	>50K

Figure 2.1: The original dataframe

We took inspiration for the pre-processing from the AIF360 `load_preproc_data_adult`, which is an adaptation from https://github.com/fair-preprocessing/nips2017/blob/master/Adult/code/Generate_Adult_Data.ipynb

We tried adding several other features, and that increased the accuracy, but while for some attributes like capital gain, capital loss or hours per week, this was just beneficial, when adding marital status or relationship, an impact on the fair classifier of the next chapter could have been noticed.

In conclusion, we actually added just the feature *Hours per week* as the goal of the project was not to find the best classifier, but rather to show how fairness and privacy can affect a classifier.

Also, we have binarized the age into two class, one which include the ages between 35 and 55 and the second representing what was left.

We later splitted the data set into train, validation and test, using the proportion 70 : 15 : 15

2.3 Model

As a model we decided to use a Decision Tree. In fact, different models were tried, but there was no drastic improvement in precision. The decision tree returns the probability that a certain person has a high or low income.

We then used the validation set to find the best threshold for the probability after which we could consider an element of a class or of another. Approximately, by running the whole process different times, we found that a good threshold is around 0.25.

2.4 Result

We then used the so found best threshold to classify the test data set. What we obtained is a balanced accuracy of around 0.72. For balanced accuracy we mean the average of the true positive and negative rate.

$$BA = \frac{TNR + TPR}{2}$$

3.1 Overview

Assess the group fairness of the classifier, assuming the protected attributes are Age, Sex. Choose any fairness metric, and apply a technique to ensure the classifier is fair. We will refer to it as the **fair classifier**. Report the chosen fairness metric on the classifier and on the fair classifier.

3.2 Privileged groups

As multiple protected attributes are being considered, we have to understand first what privileged means. The attributes we want to protect are race and age. We considered privileged race, the value 'White' which in our pre-processing will be transformed into 1. All the other races will be assigned to 0.

For the second attribute we decided to identify a privileged age, in the range between 35 and 55, as we thought it is the age where a person may earn the most. Hence, as said before we binarized the age attribute using this rationale.

As it can be noticed from 3.1 we decided to actually create 3 different classes:

- Very privileged: tuples of this class both have the privileged race, and are in their privileged age.
- Slightly privileged: tuples of this class have one of the two protected attribute, assigned to a privileged value and the other to the unprivileged class.
- Unprivileged: these tuples do not have neither a privileged age nor a privileged race.

As the following algorithms do not accept more than two classes, we actually create a fourth more generic class, called privileged, where we merge both the very and slightly privileged tuples.

3.3 Fair Classifier

To ensure fairness we have decided to use the Reweighting method, which will assign different weights to tuples of the different above defined classes, in particular the unprivileged and privileged groups as defined

```

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

```

Figure 3.1: Privileged groups

in 3.2.

```

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

```

Figure 3.2: Before and after reweighing

Defined the difference of mean outcomes as

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

We then logged the difference in mean outcomes for all the three of the privileged classes defined, obtaining of course a mean of zero, for the group we used as reference, and also having a value closer to 0 for the other two classes.

With this method, we actually ended up penalizing the privileged groups, as the difference in mean outcomes for the slightly and very privileged classes are now greater than 0, but because it is still close to 0, we have decided to keep this solution.

Reweighting is not the only algorithm to ensure fairness, there are other that AIF360 offers, like DisparateImpactRemover, LFR and OptimPreproc. But, the first seems to work best with only one protected attributes, and the other two, required some hyper parameters to efficiently work.

To keep our study simple we decided to stick with the Reweighting algorithm, which as we will see offer great results in many of the fairness metrics we will focus on.

3.4 Result

Using the same procedure described in the classification of chapter 1, we obtain a classifier with a quite lower accuracy.

At the same time the fairness measures have a great improvement over the basic classifier, as showed in figure 3.3

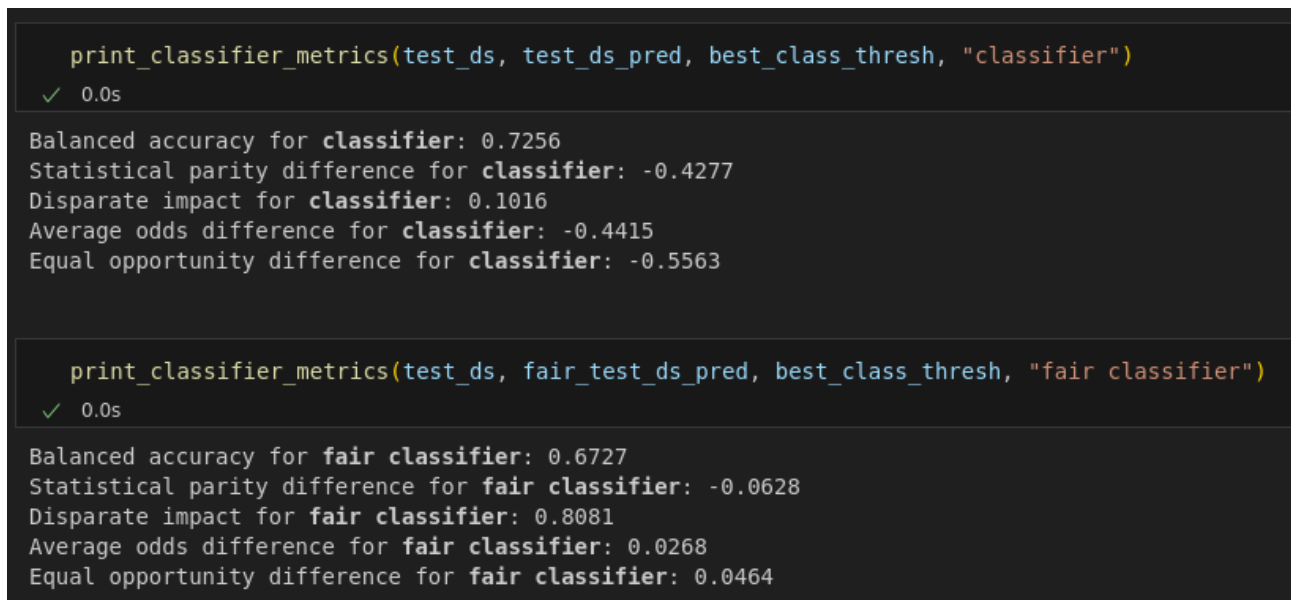


Figure 3.3: Fair classifier vs classifier

- The Statistical Parity Difference, which is a synonym of what we have previously defined as difference in mean outcomes is now closer to 0, meaning it improved.
- The Disparate Impact, which is $\frac{Pr(Y=1|D=\text{unprivileged})}{Pr(Y=1|D=\text{privileged})}$ also improved as it is now closer to 1, which would be its optimal value.
- The Average Odds Difference, which is defined as the average of difference in false positive rate(FPR) and true positive rate(TPR) for unprivileged and privileged groups is now closer to 0, the value that would assure equality in the odds.
- The Equal Opportunity Rate, which is defined as the difference of the true positive rates for the unprivileged and privileged groups, is also now closer to 0.

Coming back to what we had hinted in the first chapter, in case we added the relationship or the marital status feature, in our data set, and we repeated the exact same process described until now, the two new classifiers will have results looking like figure 3.4.

```
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
```

Figure 3.4: Classifiers with marital status

We can notice a great improvement in both the classifiers regarding the accuracy, but now the Reweighing, only slightly improved the fairness measures. We suspect this is due to the probable connection between the marital status feature and the protected attribute age. For example, a reason could be the fact that marital-status is a highly predictive feature for the classifiers, meaning they strongly rely on it. And because of its correlation with the protected attribute age, the reweighing algorithm may struggle counteracting the unfairness of the dataset.

4.1 Overview

Assume that the attributes Age, Sex are sensitive. Compute a cross-tabulation showing how many people exist in value combinations of the two sensitive attributes.

Apply local differential privacy to the responses of the individuals about what is their Age and Sex, and create a private data set. You may want to explore various epsilon values. Compute a cross-tabulation on the private data, and estimate how many people exist in value combinations of the two sensitive attributes. Quantify the errors in the estimation. Split the private data in the same manner as in (1), and train a classifier; we will refer to it as the **private classifier**. Measure the performance of the private classifier. Is there an impact on model performance due to privacy compared to the classifier?

4.2 Initial Exploration

For section 3 we part from the assumption that age and sex are sensitive attributes in our dataset. This means that they will require some amount of processing to protect the privacy of the people who's information we're working with. We begin by performing a cross-tabulation for these two attributes to see how many people fall within each combination. For this cross tabulation we work with the binary age value calculated in exercise 1. As we can see the dataset is predominantly male and outside the privileged age range

sex	Female	Male	All
age			
0.0	6908	12525	19433
1.0	3863	9265	13128
All	10771	21790	32561

Figure 4.1: Cross-tabulation of initial values

defined in exercise 1.

4.3 Applying Local Differential Privacy

In order to ensure the privacy of the people in the database regarding the two sensitive attributes we need to apply local differential privacy (LDP). We chose a local model instead of a central privacy model because we cannot assume that there is a trusted data curator who will properly execute differentially private mechanisms before sending the data to an analyst.

The method chosen to apply LDP was a randomized response algorithm. We implement the algorithm as well as create functions to generate P and Q values used by the algorithm based on an Epsilon value. This will allow us to easily see how the privatize data reacts to different Epsilon values.

```
1 import random
2 import math
3
4 def rand_resp(x, p, q):
5     toss = random.random()
6     if x == 0:
7         y = 0 if toss <= q else 1
8     else:
9         y = 1 if toss <= p else 0
10    return y
11
12 # Randomized response implementation
13 def apply_local_privacy(df, p, q):
14
15     df['priv_sex'] = df['sex'].apply(lambda x: rand_resp(x, p, q))
16
17     #Binarize age first before applying local privacy
18     df['age'] = df['age'].apply(lambda x: is_in_privileged_age(x))
19     df['priv_age'] = df['age'].apply(lambda x: rand_resp(x, p, q))
20
21 # P and Q value generator for a specific epsilon value
22 def get_p_q(epsilon):
23     p = math.exp(epsilon)/(1+math.exp(epsilon))
24     return p, p
25
26 # Apply random response to the dataframe with chosen P and Q values
27 def apply_rand_resp(truth, p=0.75, q=0.75):
28     return np.array([rand_resp(x, p, q) for x in truth])
```

Code 4.1: Randomized Response Implementation

In order to verify that the privacy algorithm doesn't significantly affect the value distribution we'll check the percent of males and privileged age people in the dataset before and after applying LDP.

```
1 priv_age = df['age'].apply(lambda x: 1 if x == 1 else 0).values
2 males = df['sex'].apply(lambda x: 1 if x == 'Male' else 0).values
3
4 n_priv_age = np.sum(priv_age)
5 n_males = np.sum(males)
6 n_people = len(priv_age)
```

Code 4.2: Estimate Function

We also implement an estimate function to compare how the privatized data compares to the non-privatized dataset.

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

Code 4.3: Estimate Function

Finally we use the implemented functions in a loop checking epsilon values from 1 to 3 in .25 increments.

```
1  epsilons = [1, 1.25, 1.5, 1.75, 2, 2.25, 2.5, 2.75, 3]
2  for x in epsilons:
3      epsilon = x
4      p, q = get_p_q(epsilon)
5      print(f"For {epsilon:.3f}-LDP we set p={p}, q={q}.")
6
7      priv_age_responses = apply_rand_resp(priv_age, p, q)
8      n_est_priv_age = estimate(priv_age_responses, p, q)
9
10     error = round(abs((n_priv_age - n_est_priv_age)/(n_priv_age) * 100), 2)
11
12     print("-----")
13     print(f"There is an estimated {n_est_priv_age:.0f} people of privileged age.")
14     print(f"This is very close to the actual number of {n_priv_age} people of
15     privileged age.")
16     print(f"With and error of {error}%\n")
17
18     male_responses = apply_rand_resp(males, p, q)
19     n_est_males = estimate(male_responses, p, q)
20     error = round(abs((n_males - n_est_males)/(n_males) * 100), 2)
21
22     print(f"There is an estimated {n_est_males:.0f} males.")
23     print(f"This is very close to the actual number of {n_males} males.")
24     print(f"With and error of {error}%\n\n")
```

Code 4.4: Epsilon Value Comparison

Fig 4.2 shows some of the results of the run.

As we can see the change in epsilon values, and the corresponding P and Q values affect how much the values with LDP applied differ from the values with no privacy applied. Looking at the values we chose an Epsilon of 2.750 and the associated 0.939 value for both P and Q to ensure the least amount of error.

We also perform another cross-tabulation to check how the LDP affected values compare to the original ones.

4.4 Classifier Integration

For the following part we integrated our LDP algorithm into the classifier created for exercise 1.

For 2.250-LDP we set $p=0.9046505351008906$, $q=0.9046505351008906$.

There is an estimated 13028 people of privileged age.
This is very close to the actual number of 13128 people of privileged age.
With and error of 0.76%

There is an estimated 21871 males.
This is very close to the actual number of 21790 males.
With and error of 0.37%

For 2.500-LDP we set $p=0.9241418199787564$, $q=0.9241418199787564$.

There is an estimated 13065 people of privileged age.
This is very close to the actual number of 13128 people of privileged age.
With and error of 0.48%

There is an estimated 21762 males.
This is very close to the actual number of 21790 males.
With and error of 0.13%

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 privileged 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 privileged 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%

Figure 4.2: Results of Epsilon Value Analysis

priv_sex	0	1	All
priv_age			
0	2408	27516	29924
1	221	2416	2637
All	2629	29932	32561

Figure 4.3: Cross-tabulation With LDP Applied

```

1 def load_preproc_data_adult(protected_attributes=None):
2     min_privileged_age = 35
3     max_privileged_age = 55
4     p = 0.88
5     q = 0.88
6     def custom_preprocessing(df):
7
8         def is_in_privileged_age(x):
9             if x > min_privileged_age and x < max_privileged_age:
10                 return 1.0
11             else:
12                 return 0.0
13
14         def group_edu(x):
15             if x <= 5:
16                 return '<6'
17             elif x >= 13:
18                 return '>12'
19             else:
20                 return x
21
22         def group_race(x):
23             if x == "White":
24                 return 1.0
25             else:
26                 return 0.0
27
28         def rand_resp(x, p, q):
29             toss = random.random()
30             if x == 0:
31                 y = 0 if toss <= q else 1
32             else:
33                 y = 1 if toss <= p else 0
34             return y
35
36         # Limit education range
37         df['education years'] = df['education-num'].apply(lambda x: group_edu(x))
38         df['education years'] = df['education years'].astype('category')
39
40         # Rename income variable
41         df['income binary'] = df['income-per-year']
42         df['income binary'] = df['income binary'].replace(to_replace='>50K.', value='
>50K', regex=True)

```

```

43     df['income binary'] = df['income binary'].replace(to_replace='<=50K.', value=
'<=50K', regex=True)
44
45     # Recode sex, race and binarize age
46     df['sex'] = df['sex'].replace({'Female': 0.0, 'Male': 1.0})
47     df['race'] = df['race'].apply(lambda x: group_race(x))
48     df['age'] = df['age'].apply(lambda x: is_in_privileged_age(x))
49
50     # Apply local privacy to sensitive values
51     df['sex'] = df['sex'].apply(lambda x: rand_resp(x, p, q))
52     df['age'] = df['age'].apply(lambda x: rand_resp(x, p, q))
53
54     return df
55
56
57     XD_features = ['age', 'education years', 'sex', 'race', 'hours-per-week']
58     D_features = ['age', 'race', 'sex'] if protected_attributes is None else
protected_attributes
59     Y_features = ['income binary']
60     X_features = list(set(XD_features)-set(D_features))
61     categorical_features = ['education years']
62
63     # privileged classes
64     all_privileged_classes = {"age": [1.0],
65                               "race": [1.0],
66                               "sex": [1.0]}
67
68     # protected attribute maps
69     all_protected_attribute_maps = {"age": {1.0: f'Between {min_privileged_age} and {
max_privileged_age}', 0.0: f'Not in between {min_privileged_age} and {
max_privileged_age}'},
70                                     "race": {1.0: 'White', 0.0: 'Non-white'},
71                                     "sex": {1.0: 'Male', 0.0: 'Female'}}
72
73     return AdultDataset(
74         label_name=Y_features[0],
75         favorable_classes=['>50K', '>50K.'],
76         protected_attribute_names=D_features,
77         privileged_classes=[all_privileged_classes[x] for x in D_features],
78         instance_weights_name=None,
79         categorical_features=categorical_features,
80         features_to_keep=X_features+Y_features+D_features,
81         na_values=['?'],
82         metadata={'label_maps': [{1.0: '>50K', 0.0: '<=50K'}],

```

```
83         'protected_attribute_maps': [all_protected_attribute_maps[x]  
84                                     for x in D_features]}},  
85     custom_preprocessing=custom_preprocessing)
```

Code 4.5: Classifier With LDP

Running the classifier with the same 70/30 split used in exercise 1 we obtain an accuracy of 0.7094. This value is lower than the value obtained in exercise 1 which shows that model performance is negatively affected by the LDP implementation.

5.1 Overview

Consider the same fairness metric and fairness mitigation method as in (2). Create a fair version of the private classifier; we will refer to it as **private+fair classifier**.

Assume, you're an auditor that has access to the real sensitive values of Age and Sex. Using the real values of Age and Sex, measure the fairness of the private+fair classifier, and compare it to that of the fair classifier. Draw conclusions.

5.2 Methodology

1. Fairness Mitigation (Reweighting):

- The dataset was reweighted to address fairness concerns. Reweighting assigns weights to instances such that the dataset compensates for historical biases and aligns with fairness objectives.
- Privileged and unprivileged groups were defined based on combinations of **Sex** and **Age**:
 - **Very Privileged Groups:** sex=1, age=1
 - **Slightly Privileged Groups:** sex=0, age=1, sex=1, age=0
 - **Unprivileged Groups:** sex=0, age=0

2. **Dataset Splitting:** The reweighted dataset was split into training, validation, and test sets (70% training, 15% validation, and 15% test).

3. Classifier Training:

A **Decision Tree Classifier** was trained on the reweighted dataset using instance weights to reflect the adjusted importance of samples.

4. Evaluation Metrics:

- **Fairness Metrics:** Statistical parity difference, disparate impact, equal opportunity difference, and average odds difference.
- **Performance Metrics:** Balanced accuracy, Theil index (information distribution), and predictive performance on test data.

5.3 Results

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

Table 5.1: Comparison of Fairness and Performance Metrics between Fair Classifier and Private+Fair Classifier

- The **Fair Classifier** exhibited substantially better fairness metrics across all dimensions:
 - **Statistical Parity Difference** and **Disparate Impact** show that the Private+Fair Classifier favors privileged groups more strongly than the Fair Classifier.
 - Larger biases in **Equal Opportunity** and **Average Odds Difference** indicate unequal treatment between privileged and unprivileged groups.
- The **Private+Fair Classifier** achieved a higher balanced accuracy (72.1%), indicating better predictive performance.
- The improved **Theil Index** (0.115) suggests a more even distribution of information.
- The **Private+Fair Classifier** sacrifices fairness to improve performance. This trade-off poses ethical and regulatory challenges, particularly in high-stakes domains such as hiring, credit allocation, or healthcare.

5.4 Recommendations

- **Context-Specific Use:** The Private+Fair Classifier may be suitable in contexts where performance is paramount and fairness is less critical. For applications prioritizing equity, the Fair Classifier is preferable.
- **Fairness-Performance Balance:** Consider alternate mitigation techniques (e.g., adversarial debiasing or post-processing adjustments) to strike a balance between fairness and accuracy.

- **Transparency and Monitoring:** Employ fairness auditing and regular monitoring to ensure compliance with ethical guidelines.

6.1 Overview

Study the explainability of the private classifier. Identify instances where the model is wrong but highly confident, and explain them. Assume you have access to the real sensitive values of Age and Sex.

Investigate whether the noisy values for these attributes are responsible for the model being confident and wrong.

6.2 Methodology

- **Threshold:** Firstly, we defined a threshold that will help us to identify any classification prediction with a confidence score equal or greater than 0.9. The predictions will be considered as **high confidence**.

```
1 high_confidence_threshold = 0.9
2
```

Code 6.1: Threshold Definition

- **Predicted Labels:**

By creating a new variable where compares if the predicted scores are higher than the best class threshold.

```
1 predicted_labels = (test_ds_pred.scores > best_class_thresh).astype(int)
2
```

Code 6.2: Predicted labels Definition

- **Actual Labels:**

In this part of the code, we needed to store the accurate labels for a latter comparison:

```
1 actual_labels = test_ds.labels.ravel()
```

2

Code 6.3: Actual labels Definition

- **High-Confidence Errors:**

Once that we defined our labels, in this part, we compare if the predicted labels are not the same as the actual ones. Additionally, it makes sure that the scoring is higher than our pre-defined threshold (0.9).

- **Result:**

Finally, we show an array containing the indices of a misclassified example in the set where the prediction had a confidence score greater than 0.9:

```
1 errors_with_high_confidence
2
```

Code 6.4: Errors with high confidence

6.3 Results

The result:

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

shows a large number of repeated errors, suggesting that the model fails with high confidence in many specific cases. The model may be over fitting or relying too much on certain features that do not generalize well.

- **Errors concentrated in certain characteristics:** The variables age, race, and sex are binarized: age (0.0 or 1.0), race (1.0 meaning "White"), and sex (1.0 meaning "Male"). Many instances show age = 0.0 and sex = 1.0, indicating that they might be young individuals who are not in the age-privileged class and are male. This suggests that the errors occur primarily in specific subgroups, such as young, age-nonprivileged males.
- **Homogeneous distribution of non-differentiating features:** Most rows have hours-per-week = 40.0, and in "education years", only one category has a value of 1, the others are 0.0, so the model could be ignoring other important features, focusing too much on a few, which reduces its ability to capture general patterns.
- **High but incorrect confidence:** All instances show predictions with high confidence (≥ 0.6), which is concerning as these predictions are incorrect. The model might be overestimating its confidence in certain scenarios where key features, such as age and sex, are noisy or not well represented.

- **Possible issue with noise in the data:** Preprocessing that introduces noise in age and sex could be responsible for these erroneous predictions. The noise added for privacy may be disproportionately affecting certain groups, causing systematic errors in these categories.

These errors indicate that the private model does not handle certain groups well, which could be due to noise in sensitive features. It is recommended to evaluate noise reduction in age and sex or adjust hyper parameters to improve the balance between privacy and accuracy.

7.1 Overview

Select an explainability method and create a natural language interface to it. The idea is to take an explanation (e.g., feature importance pairs, examples) and present it as text that a person can easily understand.

7.2 Methodology

For this part of the project, the LIME explainability method was chosen, as it gives explanations for individual predictions, which is helpful when looking at high-confidence errors. Other methods like SHAP or global feature importance focus more on overall trends, but LIME lets us see why the model was confidently wrong for specific cases in the previous task. This shall help us to find potential patterns in the errors, figure out which features might be causing problems, and possibly to decide how to improve the model or the data preprocessing. The resulting explanations of the model are then provided to a Large Language Model to be explained in natural language. In this case, we will be analysing value pairs consisting of attributes values and their thresholds.

The model uses the "errors with high confidence" output from the previous task as its input. This means that we use a list of indices representing instances where the model was highly confident but made incorrect predictions. Due to the extended run-time of the code, we chose to limit the use of "errors with high confidence" to 10.

The following code shows the steps taken to implement the LIME explainability method, starting with the installation of the methodology.

```
1 !pip install lime
2
```

Code 7.1: Install LIME as chosen explainability method

In the following first part of the code the `LimeTabularExplainer` is imported, a class from the LIME library specifically designed to explain predictions for tabular datasets. Afterwards we initialize the explainer. We

set the training data set as X-train, and our training labels as y-train, as well as all other features used in the model, like "age" or "education". We also set the classification attribute and range and declare the treating of continuous attributes, like "age" or "education" to be discretized into bins.

```
1 from lime.lime_tabular import LimeTabularExplainer
2 import json
3
4 explainer = LimeTabularExplainer(
5     X_train,
6     training_labels=y_train,
7     feature_names=test_ds.feature_names,
8     class_names=["<=50K", ">50K"],
9     discretize_continuous=True
10 )
11
```

Code 7.2: Actual labels Definition

In the second half of the models execution, we run through the list of "error with high confidence". For presentation purposes and runtime limitations, the model only runs through the first 30 errors and list 5 features per incident. This generates an explanation for the X-test[idx], showing which features contributed the most to the predictions.

```
1 subset_errors = errors_with_high_confidence[:30]
2
3 explanations = {}
4 for idx in subset_errors:
5     instance = X_test[idx].reshape(1, -1)
6     exp = explainer.explain_instance(
7         instance.flatten(),
8         dtmod.predict_proba,
9         num_features=5
10     )
11     explanations[idx] = exp.as_list()
12
```

Code 7.3: Scan through top 10 errors with high confidence

The following code prints the models results in a JSON File.

```
1 explanations_serializable = {
2     int(key): [f"{feature}: {contribution}" for feature, contribution in value]
3     for key, value in explanations.items()
4 }
5 with open("lime_explanations.json", "w") as json_file:
6     json.dump(explanations_serializable, json_file, indent=4)
```

Code 7.4: Convert LIME explanations to a serializable format Definition

Next the models output is printed out as a .json file to be readable.

```

1 {
2   "0": [
3     "education years=>12 <= -0.57: -0.31912072442281514",
4     "hours-per-week <= -0.03: -0.16427014072192786",
5     "education years=11 > -0.21: 0.10027709197379227",
6     "education years=12 <= -0.19: -0.086096599234411",
7     "age <= -0.86: -0.058996254575927914"
8   ],
9   "1": [
10    "education years=>12 > -0.57: 0.3168093711193335",
11    "-0.03 < hours-per-week <= 0.37: 0.12681860011219673",
12    "education years=11 <= -0.21: -0.11116524737044363",
13    "-0.86 < age <= 1.16: 0.06644305258389807",
14    "education years=10 <= -0.53: -0.04891011018552507"
15  ]
16 }
17

```

Code 7.5: Output .JSON file

7.3 Results

To finalise this part of the project and to get a better understanding of the models results, the resulting .JSON File was given to the Large Language Model "Llama-3.2-1B-Instruct-Q8-0-GGUF" via the Software "LM Studio" with configurations depicted in the following screenshot.

In the last step, the the .JSON file is handed to the LLM, with the prompt: "Could you explain the following .json file to content in easy understandable language and only focusing on the key aspects within 2 or 3 paragraphs"

The LLM came up with the following explanation:

"I'd be happy to break down the JSON file for you. Here's what it means:

This is a JSON (JavaScript Object Notation) file that contains information about educational requirements. In simple terms, it says what education level is required for each job in different age groups and hours/week. The requirements are divided into three categories: 12 years old or younger, 13-19 years old, and 20 years old or older.

For example, if you want to know the minimum education requirements for a job that pays \$50,000 per year (which is relatively high), it might say that an individual under the age of 12

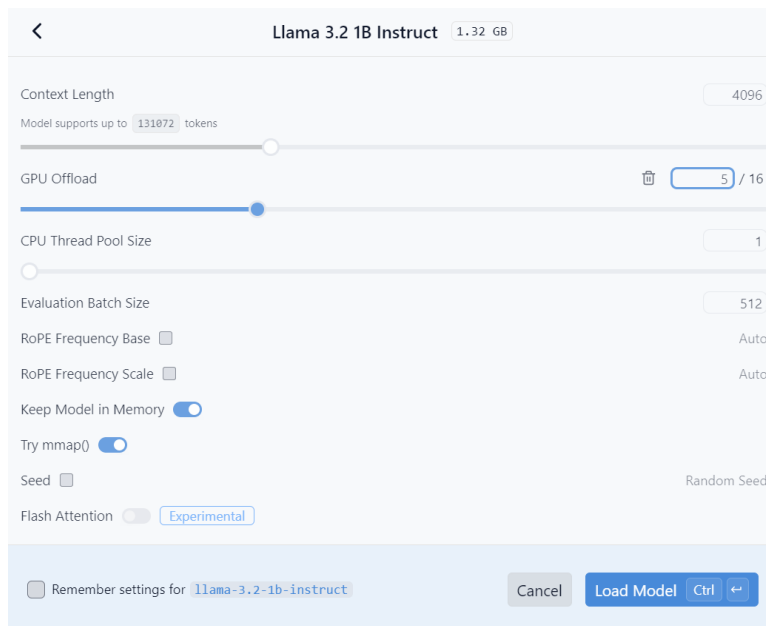


Figure 7.1: LLM-Configurations

needs to have at least 7 years of education. On the other hand, someone who works full-time in their mid-30s may need only 11 years of education to reach a similar salary.

The JSON file provides a list of key requirements for different age groups and hours/week, which can be broken down into various categories such as education years (e.g. 12 is required), hours per week (e.g. under 0 is not recommended), and age (e.g. less than -0.86 means you're under a certain threshold).

When asked to elaborate in further detail, a more detailed analysis of the data is constructed.

Let's break down the JSON file further to understand its contents better.

Overview The JSON file appears to be an explanation of the minimum education requirements for various jobs across different age groups and hours/week. It provides specific details on what education level is required, how long it should last, and other relevant factors.

Structure

The JSON file consists of a single object with two properties: 0 and 1. These objects represent two separate categories or branches of information. Each branch (0 and 1) has an array of key-value pairs. These key-value pairs are further divided into three subcategories: education years=>12 <= -0.57: This indicates that for jobs with a minimum education requirement of 12 years, the corresponding value is -0.31912072442281514. This suggests that having at least 13 years of education is required to meet this threshold. Similarly, there are multiple subcategories under hours-per-week <= -0.03: and other similar conditions for other age groups (e.g., age <= -0.86: -0.058996254575927914'). Each key-value pair in the array represents a specific condition or requirement. **Example**

To illustrate this, let's consider an example:

For jobs with a minimum education requirement of 12 years under age 13, the value might be 0.10027709197379227. This indicates that having at least 11 years of education (with a grade level equivalent to 11) is required for these jobs. The same logic applies to other age groups and hours/week conditions. Key Takeaways

To summarize:

Education Requirements: The JSON file provides detailed information on the minimum education requirements for various jobs across different age groups and hours/week. **Age Groups:** There are separate branches for each age group, with specific values indicating the corresponding minimum education requirement. **Hours/Week Conditions:** Similarly, there are multiple conditions under which certain education levels are required. I hope this explanation helps you understand the contents of the JSON file! Let me know if you have any further questions.

To conclude the results of the used explainability method LIME, show some interesting insights, that could not be properly understood and explained by the used LLM without further context. But regarding the results of the model alone, we can conclude, that according to the model the years of education, Hours per week and also the chosen privileged age group play a key role in determining the outcome of the classifier.

8.1 Overview

Play around with the dataset and report any interesting (in terms of responsible data science) insights that you may find.

8.2 Income by age

The first graph shows how the various age groups are distributed, considering a salary greater than 50K per year. In particular, it is observed that the age group that represents the largest slice of the pie chart is the one that corresponds to people ranging between 25-34, with a percental of 26.6%, right after by a small percentage gap, instead there is the age group 35-44 with 26.3% . Instead, as might have been expected, only 4.8 percent of people over 65 can guarantee higher earnings than 50K.

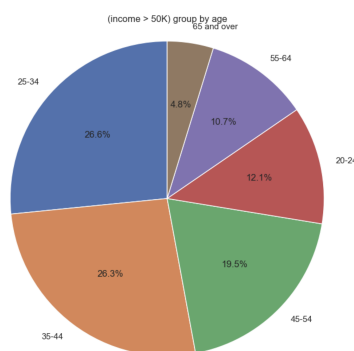


Figure 8.1: Pie chart by Income and Age

8.3 Income by education and work

Below is a chart that intuitively shows the correlation between academic level and the type of work performed by individuals in the 'Adult' dataset. It is immediately clear that the private sector has a peak well

above 1400, corresponding to the Bachelor's degree. In general, almost all sectors show a peak at the Bachelor's level of education. This leads to the intuitive observation that, currently, for all jobs, the majority of people prefer to obtain a bachelor's degree.

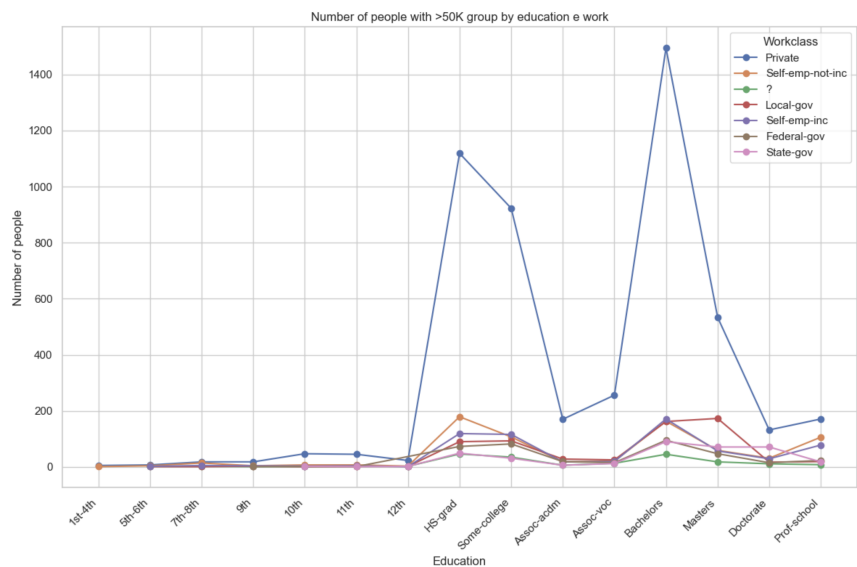


Figure 8.2: Graph of Income group by education and work

8.4 Income by gender

The following charts show the number of individuals, both for those with an average salary of <50K and those with a salary above 50K, for each industry, broken down by gender.

It can be observed that, in the private sector, the number of males earning more than 50K is significantly higher compared to females, with a gap of 4000. This gap reduces considerably when looking at people earning less than 50K. In general, in every sector, the male population exceeds the female population with a significant gap when considering salaries greater than 50K.

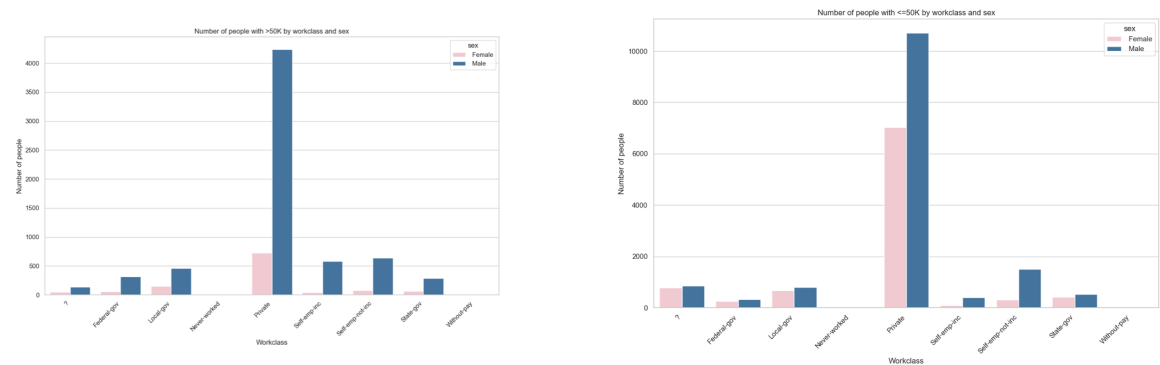


Figure 8.3: Graph of income by gender

8.5 Income by gender and education

However, if we look at the graphs that analyze the education level by gender, broken down by salary, we can observe a significant gap at the Bachelor's level between the two sexes, of almost 1300, for salaries greater than 50K. This gap is considerably smaller when considering salaries $\leq 50K$, as the difference is less than 500. In general, this clear disparity can be attributed to the fact that, as seen in the second chart, individuals who have at least completed a Bachelor's degree have a higher likelihood of earning a salary above 50K.

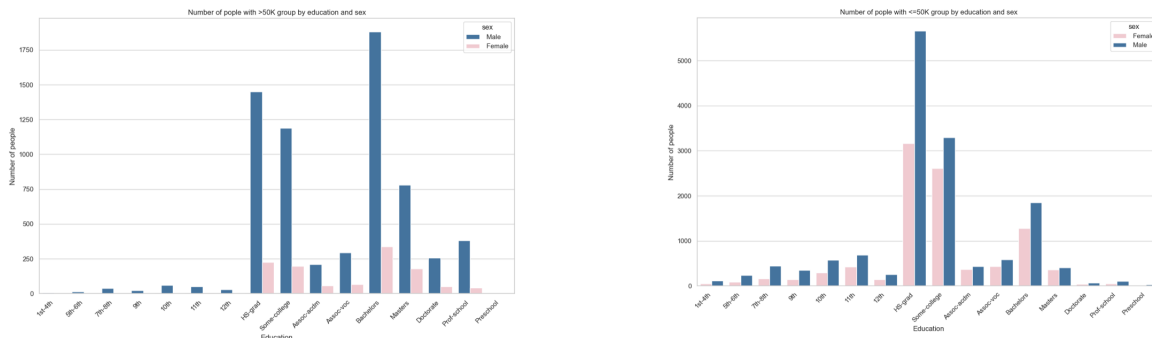


Figure 8.4: Graph of income by gender and education

8.6 Income by Education year

By conducting a more specific analysis to identify the column most closely related to a salary greater than 50K, it can be observed that the relevant column is 'Edu-num.' It is evident that individuals who have completed at least 13 years of education (equivalent to a high school diploma or higher) represent a higher number of people earning more than 50K.

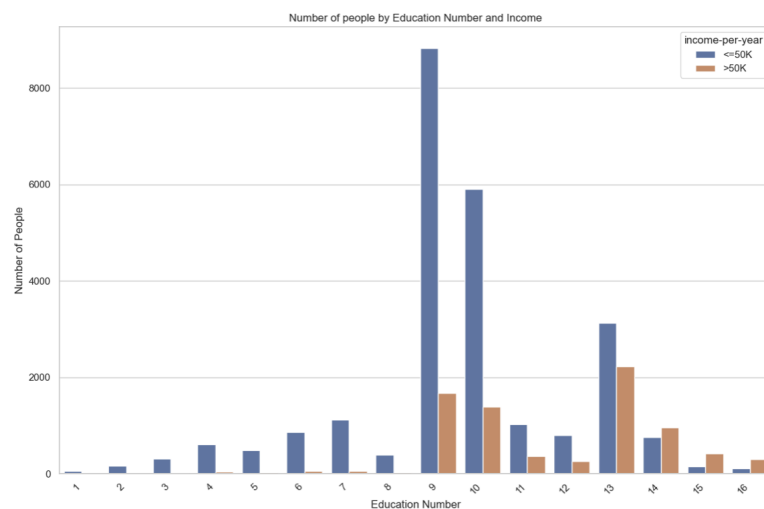


Figure 8.5: Graph of income by education years

In the classification, we applied some pre-processing to our data, by binarizing the age, adding some other features and we trained a Decision Tree.

We later explored the concept of fairness, by marking age and race as protected. We showed that Reweighting managed to improve the fairness metrics, although it also lowered down the accuracy of the model. Lastly we showed the impact of adding other features like marital status on fairness.

In terms of privacy we can see that using the random response algorithm it's easy to implement local differential privacy in a way that does not significantly impact the overall composition of our dataset. Nonetheless, the integration of LDP does have a negative impact in the classifier model.

Although the Private+Fair Classifier demonstrates superior accuracy, its increased disparity in fairness metrics highlights significant concerns. Organizations must weigh the implications of these trade-offs to align model deployment with ethical and operational goals.

As for explainability, the errors indicate that the private model does not handle certain groups well, which could be due to noise in sensitive features. It is recommended to evaluate noise reduction in age and sex or adjust hyper parameters to improve the balance between privacy and accuracy.

Using the explainability method LIME, which generates feature-importance pairs showing how each attribute contributes to a prediction, we can gain insights into the model's highly confident but incorrect predictions. This helps us understand correlations, such as the minimum education requirements for various jobs across different age groups and weekly working hours. According to the model's outcomes, these factors significantly influence its classification decisions.

References

- [Bel+18] Bellamy, Rachel K. E. et al. (Oct. 2018). *AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias*. URL: <https://arxiv.org/abs/1810.01943>. (accessed: 1.12.2024).
- [Wan18] Wang, Graham Cormode Somesh Jha Tejas Kulkarni Ninghui Li Divesh Srivastava Tianhao (2018). *Privacy at Scale: Local Differential Privacy in Practice*. URL: <http://dimacs.rutgers.edu/~graham/pubs/papers/ldptutorial.pdf>. (accessed: 1.12.2024).