Appendix 1. Jupyter Notebook

COMPAS Bias analysis

February 4, 2024

COMPAS Data exploration and initial model creation

[19]:
```
import pandas as pd

file_path = "compas-scores-two-years.csv"
compas_data = pd.read_csv(file_path)

## describing the overall dataset
print(compas_data.describe)
```

```
<bound method NDFrame.describe of          id                 name      first
last  \
0         1     miguel hernandez     miguel    hernandez
1         3          kevon dixon      kevon        dixon
2         4             ed philo         ed        philo
3         5          marcu brown      marcu        brown
4         6   bouthy pierrelouis     bouthy  pierrelouis
…       …                   …         …           …
7209  10996        steven butler     steven       butler
7210  10997      malcolm simmons    malcolm      simmons
7211  10999      winston gregory    winston      gregory
7212  11000          farrah jean     farrah         jean
7213  11001  florencia sanmartin  florencia    sanmartin

     compas_screening_date     sex         dob  age          age_cat  \
0               2013-08-14    Male  1947-04-18   69  Greater than 45
1               2013-01-27    Male  1982-01-22   34          25 - 45
2               2013-04-14    Male  1991-05-14   24     Less than 25
3               2013-01-13    Male  1993-01-21   23     Less than 25
4               2013-03-26    Male  1973-01-22   43          25 - 45
…                     …     …          …  …              …
7209            2013-11-23    Male  1992-07-17   23     Less than 25
7210            2014-02-01    Male  1993-03-25   23     Less than 25
7211            2014-01-14    Male  1958-10-01   57  Greater than 45
7212            2014-03-09  Female  1982-11-17   33          25 - 45
7213            2014-06-30  Female  1992-12-18   23     Less than 25

                  race  …  v_decile_score  v_score_text  v_screening_date  \
0                Other  …               1           Low        2013-08-14
```

```
1     African-American  …               1          Low        2013-01-27
2     African-American  …               3          Low        2013-04-14
3     African-American  …               6       Medium        2013-01-13
4                Other  …               1          Low        2013-03-26
…                    …  …             …           …                  …
7209  African-American  …               5       Medium        2013-11-23
7210  African-American  …               5       Medium        2014-02-01
7211             Other  …               1          Low        2014-01-14
7212  African-American  …               2          Low        2014-03-09
7213          Hispanic  …               4          Low        2014-06-30

      in_custody  out_custody  priors_count.1 start   end event two_year_recid
0     2014-07-07   2014-07-14               0     0   327     0              0
1     2013-01-26   2013-02-05               0     9   159     1              1
2     2013-06-16   2013-06-16               4     0    63     0              1
3            NaN          NaN               1     0  1174     0              0
4            NaN          NaN               2     0  1102     0              0
…            …            …               …   …   …   …            …
7209  2013-11-22   2013-11-24               0     1   860     0              0
7210  2014-01-31   2014-02-02               0     1   790     0              0
7211  2014-01-13   2014-01-14               0     0   808     0              0
7212  2014-03-08   2014-03-09               3     0   754     0              0
7213  2015-03-15   2015-03-15               2     0   258     0              1

[7214 rows x 53 columns]>
```

[20]:
```
## Same preprocessing as the ProPublica's analysis
compas_data = compas_data[(compas_data.days_b_screening_arrest <= 30)
                          & (compas_data.days_b_screening_arrest >= -30)
                          & (compas_data.is_recid != -1)
                          & (compas_data.c_charge_degree != 'O')
                          & (compas_data.score_text != 'N/A')]

columns_to_keep = ['sex', 'age', 'race', 'juv_fel_count', 'juv_misd_count',␣
 ↪'juv_other_count', 'priors_count', 'c_charge_degree', 'two_year_recid']
compas_data = compas_data[columns_to_keep]

print(compas_data.head)
```

```
<bound method NDFrame.head of          sex  age              race  juv_fel_count
juv_misd_count  \
0       Male   69             Other             0              0
1       Male   34  African-American             0              0
2       Male   24  African-American             0              0
5       Male   44             Other             0              0
6       Male   41         Caucasian             0              0
…       …  …                    …             …              …
7209    Male   23  African-American             0              0
```

```
7210    Male   23  African-American              0               0
7211    Male   57             Other              0               0
7212  Female   33  African-American              0               0
7213  Female   23          Hispanic              0               0

      juv_other_count  priors_count c_charge_degree  two_year_recid
0                   0             0               F               0
1                   0             0               F               1
2                   1             4               F               1
5                   0             0               M               0
6                   0            14               F               1
…                 …             …               …               …
7209                0             0               F               0
7210                0             0               F               0
7211                0             0               F               0
7212                0             3               M               0
7213                0             2               F               1

[6172 rows x 9 columns]>
```

[21]:
```
## Age grouping to the data
age_groups = [0, 18, 30, 40, 50, 60, 70, 80, 90, 100]
labels = ['0-18', '19-30', '31-40', '41-50', '51-60', '61-70', '71-80',␣
 ↪'81-90', '91-100']
compas_data['Age_category'] = pd.cut(compas_data['age'], bins=age_groups,␣
 ↪labels=labels, right=False)
compas_data = compas_data.drop(columns=['age'])
```

Lets see how many null values there are within the data

[22]:
```
compas_null_counts = compas_data.isnull().sum()

print(compas_null_counts)
```

```
sex                0
race               0
juv_fel_count      0
juv_misd_count     0
juv_other_count    0
priors_count       0
c_charge_degree    0
two_year_recid     0
Age_category       0
dtype: int64
```

Next we will explore how the target column of revicidism is distributed for different races and genders within the dataset.

[23]:
```
gender_recidivism = compas_data.groupby('sex')['two_year_recid'].value_counts().
  ↪unstack().fillna(0)
print(gender_recidivism)
```

```
two_year_recid     0     1
sex
Female           762   413
Male            2601  2396
```

Seems that the within the data the amount of recidivist is almost equal to non-recidivist in male population. On the other hand in the Female population, the amount of recidivists is almost half of the non-recidivists.

[24]:
```
race_recidivism = compas_data.groupby('race')['two_year_recid'].value_counts().
  ↪unstack().fillna(0)
print(race_recidivism)
```

```
two_year_recid       0     1
race
African-American  1514  1661
Asian               23     8
Caucasian         1281   822
Hispanic           320   189
Native American      6     5
Other              219   124
```

The amount of Asians, Hispanic, Native American and other's is relatively low. Due to the scope of this project, all races except Caucasians and African-American's are filtered out.

[25]:
```
compas_data = compas_data[compas_data['race'].isin(['Caucasian',␣
  ↪'African-American'])]
print(compas_data.head())
```

```
       sex              race  juv_fel_count  juv_misd_count  juv_other_count  \
1     Male  African-American              0               0                0
2     Male  African-American              0               0                1
6     Male         Caucasian              0               0                0
8   Female         Caucasian              0               0                0
10    Male         Caucasian              0               0                0

    priors_count c_charge_degree  two_year_recid Age_category
1              0               F               1        31-40
2              4               F               1        19-30
6             14               F               1        41-50
8              0               M               0        31-40
10             0               F               0        19-30
```

After the data is explored and preprocessed we will create the linear regression model

[26]:
```
## Importing libraries for the model creation
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler, OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression

## splitting dataset
def split_label(dataset, target_feature):
    X = dataset.drop([target_feature], axis=1)
    y = dataset[[target_feature]]
    return X, y

## Classification pipeline for RAI dashboard
def create_classification_pipeline(X):
    pipe_cfg = {
        'number_columns': X.dtypes[(X.dtypes == 'int64') | (X.dtypes ==␣
 ↪'float64')].index.values.tolist(),
        'category_columns': X.dtypes[X.dtypes == 'object'].index.values.
 ↪tolist(),
    }
    num_pipe = Pipeline([
        ('number_imputer', SimpleImputer(strategy='median')),
        ('number_scaler', StandardScaler())
    ])
    cat_pipe = Pipeline([('category_imputer',␣
 ↪SimpleImputer(strategy='constant', fill_value='?')),
    ('category_encoder', OneHotEncoder(handle_unknown='ignore', sparse=False))
    ])
    feat_pipe = ColumnTransformer([
        ('number_pipe', num_pipe, pipe_cfg['number_columns']),
        ('category_pipe', cat_pipe, pipe_cfg['category_columns'])
    ])

    pipeline = Pipeline(steps=[('preprocessor', feat_pipe),
                               ('model', LogisticRegression(random_state=0))])
    return pipeline


## Logistic regression model creation
target_feature = 'two_year_recid'
categorical_features = ['sex', 'race', 'Age_category', 'c_charge_degree']

train_data, test_data = train_test_split(compas_data, test_size=0.25,␣
 ↪random_state=42, stratify=compas_data['two_year_recid'])
```

```
X_train, y_train = split_label(train_data, target_feature)
X_test, y_test = split_label(test_data, target_feature)

pipeline = create_classification_pipeline(X_train)

y_train = y_train[target_feature].to_numpy()
y_test = y_test[target_feature].to_numpy()

model = pipeline.fit(X_train, y_train)
```

```
`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in
1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.
```

Initialize the Microsoft Responsible AI dashboard

[27]:
```
#Import libraries for the RAI dashboard
from responsibleai import RAIInsights
from raiwidgets import ResponsibleAIDashboard
from responsibleai.feature_metadata import FeatureMetadata
from raiutils.cohort import Cohort, CohortFilter, CohortFilterMethods

feature_metadata = FeatureMetadata(categorical_features=categorical_features,␣
 ↪dropped_features=[])

rai_insights = RAIInsights(model, train_data, test_data, target_feature,␣
 ↪'classification',
                           feature_metadata=feature_metadata)

rai_insights.explainer.add()
rai_insights.error_analysis.add()
rai_insights.compute()

## Cohorts for filtering different ethnicities and genders
cohort_caucasians = Cohort("Caucasian")
cohort_caucasians.add_cohort_filter(CohortFilter(method=CohortFilterMethods.
 ↪METHOD_INCLUDES, arg=["Caucasian"], column='race'))

cohort_african_american = Cohort("African-American")
cohort_african_american.
 ↪add_cohort_filter(CohortFilter(method=CohortFilterMethods.METHOD_INCLUDES,␣
 ↪arg=["African-American"], column='race'))

cohort_male = Cohort("Male")
cohort_male.add_cohort_filter(CohortFilter(method=CohortFilterMethods.
 ↪METHOD_INCLUDES, arg=["Male"], column='sex'))

cohort_female= Cohort("Female")
```

```
cohort_female.add_cohort_filter(CohortFilter(method=CohortFilterMethods.
 ↪METHOD_INCLUDES, arg=["Female"], column='sex'))

cohort_list = [cohort_caucasians, cohort_african_american, cohort_female,␣
 ↪cohort_male]

#RAI dashboard will be hosted locally
rai_dashboard = ResponsibleAIDashboard(rai_insights, cohort_list=cohort_list)
```

```
================================================================================
Causal Effects
Current Status: Generating Causal Effects.
Current Status: Finished generating causal effects.
Time taken: 0.0 min 4.509999416768551e-05 sec
================================================================================
================================================================================
Counterfactual
Time taken: 0.0 min 1.309998333454132e-05 sec
================================================================================

invalid value encountered in double_scalars
invalid value encountered in double_scalars
divide by zero encountered in double_scalars
divide by zero encountered in double_scalars
invalid value encountered in double_scalars
invalid value encountered in double_scalars
divide by zero encountered in double_scalars
divide by zero encountered in double_scalars
divide by zero encountered in log2
divide by zero encountered in log
categorical_feature keyword has been found in `params` and will be ignored.
Please use categorical_feature argument of the Dataset constructor to pass this
parameter.

================================================================================
Error Analysis
Current Status: Generating error analysis reports.
Current Status: Finished generating error analysis reports.
Time taken: 0.0 min 0.13721399998757988 sec
================================================================================
================================================================================
Explanations
Current Status: Explaining 8 features
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000173 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 70
```

```
[LightGBM] [Info] Number of data points in the train set: 3958, number of used
features: 8
[LightGBM] [Info] Start training from score -0.085706
Current Status: Explained 8 features.
Time taken: 0.0 min 0.42172509990632534 sec
================================================================================
ResponsibleAI started at http://localhost:8709
```

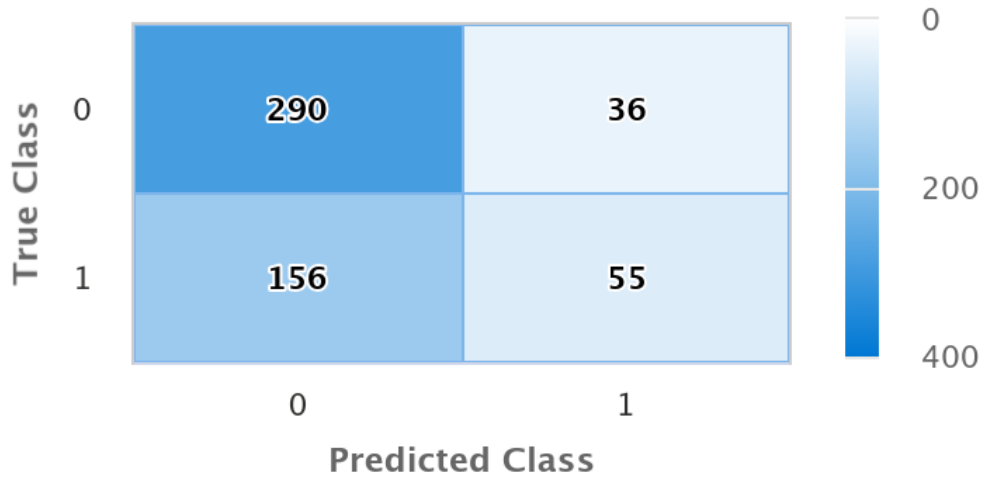
The metrics can be found in the dashboard section: Model overview by inserting features sex or race to the filter.

Race metrics:

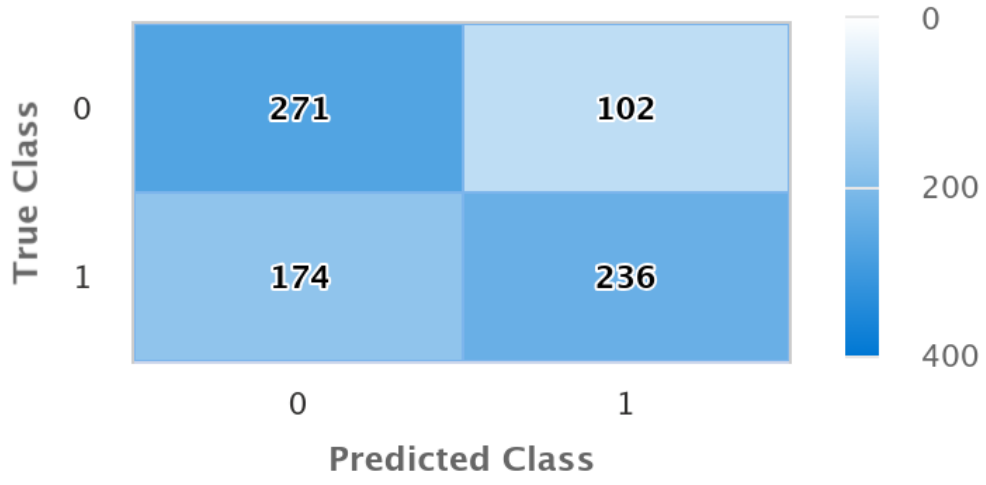
Cohorts	Sample size	Accuracy score	F1 score	False positive rate	False negative rate	Selection rat
race in African-American	783	0.648	0.631	0.273	0.424	0.432
race in Caucasian	537	0.642	0.364	0.11	0.739	0.169

Confusion matrices:

Caucasians:



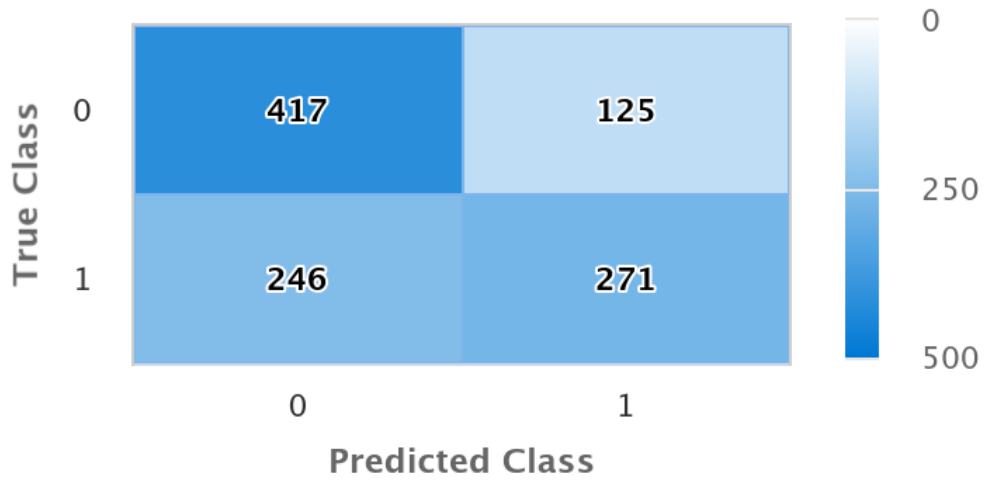
African Americans:



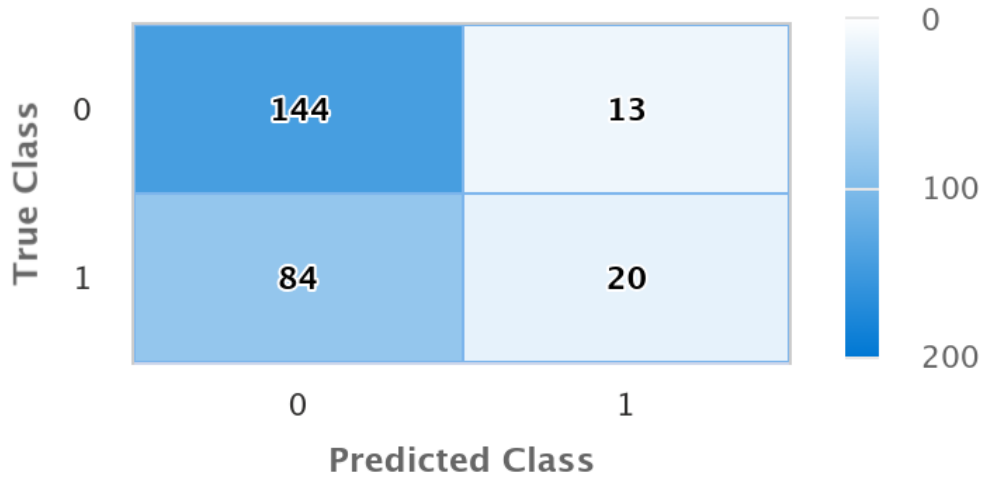
Sex metrics

Cohorts	Sample size	Accuracy score	F1 score	False positive rate	False negative rate	Selection rate
sex in Male	1 059	0.65	0.594	0.231	0.476	0.374
sex in Female	261	0.628	0.292	0.083	0.808	0.126

Males:



Females:



Based on the information provided, it seems that model does not satisfy equalized odds for neither both races or genders.

True positive and False positive rates does not match across different groups. So the model is not fair for different ethnicities or genders.

For bias mitigation we will use preprocessing method reweighin on the race and gender columns separately.

[28]:
```
## reweighing
from aif360.datasets import BinaryLabelDataset
from aif360.algorithms.preprocessing import Reweighing
from sklearn.preprocessing import LabelEncoder

train_data, test_data = train_test_split(compas_data, test_size=0.25,␣
 ↪random_state=42, stratify=compas_data['two_year_recid'])
feature_columns = train_data.columns[:-1]

## For the reweighing algorithm the data must be in binary format
sex_encoder = LabelEncoder()
c_charge_degree_encoder = LabelEncoder()
age_category_encoder = LabelEncoder()

train_data['sex'] = train_data['sex'].apply(lambda x:1 if x == 'Female' else 0)
```

```
train_data['race'] = train_data['race'].apply(lambda x:1 if x == 'Caucasian'␣
 ↪else 0)
train_data['c_charge_degree'] = train_data['c_charge_degree'].apply(lambda x:1␣
 ↪if x == 'F' else 0)
train_data['Age_category'] = age_category_encoder.
 ↪fit_transform(train_data['Age_category'])

privileged_groups = [{'race': 1}]  # Caucasians
unprivileged_groups = [{'race': 0}]  # African Americans

dataset = BinaryLabelDataset(
    favorable_label=0,
    unfavorable_label=1,
    df=train_data,
    label_names=['two_year_recid'],
    protected_attribute_names=['race'],
    unprivileged_protected_attributes=[0]
)

reweighing = Reweighing(unprivileged_groups=unprivileged_groups,␣
 ↪privileged_groups=privileged_groups)
compas_data_reweight = reweighing.fit_transform(dataset)

train_data = pd.DataFrame(data=compas_data_reweight.features, columns=dataset.
 ↪feature_names)
train_data['two_year_recid'] = compas_data_reweight.labels.ravel()


#After reweighing we will transform the data back from binary
train_data['sex'] = train_data['sex'].map({0.0: 'Male', 1.0: 'Female'})
train_data['race'] = train_data['race'].map({1: 'Caucasian', 0:␣
 ↪'African-American'})
train_data['c_charge_degree'] = train_data['c_charge_degree'].map({1.0: 'F', 0.
 ↪0: 'M'})
train_data['Age_category'] = age_category_encoder.
 ↪inverse_transform(train_data['Age_category'].astype(int))
```

[29]:
```
## retrain a model with reweighted data
X_train, y_train = split_label(train_data, target_feature)
X_test, y_test = split_label(test_data, target_feature)

pipeline = create_classification_pipeline(X_train)

y_train = y_train[target_feature].to_numpy()
y_test = y_test[target_feature].to_numpy()
```

```
model = pipeline.fit(X_train, y_train,␣
 ↪model__sample_weight=compas_data_reweight.instance_weights)

### RAI dashboard
feature_metadata = FeatureMetadata(categorical_features=categorical_features,␣
 ↪dropped_features=[])

rai_insights = RAIInsights(model, train_data, test_data, target_feature,␣
 ↪'classification',
                           feature_metadata=feature_metadata)

rai_insights.explainer.add()
rai_insights.error_analysis.add()
rai_insights.compute()

rai_dashboard = ResponsibleAIDashboard(rai_insights, cohort_list=cohort_list)
```

```
================================================================================
Causal Effects
Current Status: Generating Causal Effects.
Current Status: Finished generating causal effects.
Time taken: 0.0 min 0.00020659994333982468 sec
================================================================================
================================================================================
Counterfactual
Time taken: 0.0 min 9.599956683814526e-06 sec
================================================================================

`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in
1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

================================================================================
Error Analysis
Current Status: Generating error analysis reports.
Current Status: Finished generating error analysis reports.
Time taken: 0.0 min 0.1088529999833554 sec
================================================================================
================================================================================
Explanations
Current Status: Explaining 8 features
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000190 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 70
[LightGBM] [Info] Number of data points in the train set: 3958, number of used
features: 8
[LightGBM] [Info] Start training from score -0.097719
```

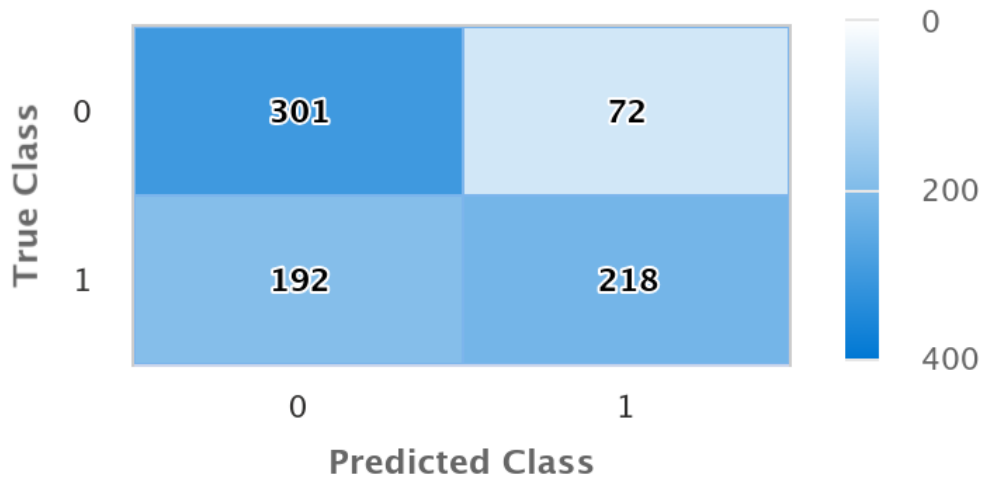
```
categorical_feature keyword has been found in `params` and will be ignored.
Please use categorical_feature argument of the Dataset constructor to pass this
parameter.

Current Status: Explained 8 features.
Time taken: 0.0 min 0.3607378000160679 sec
================================================================================
ResponsibleAI started at http://localhost:8710
```

The metrics after reweighing for different races

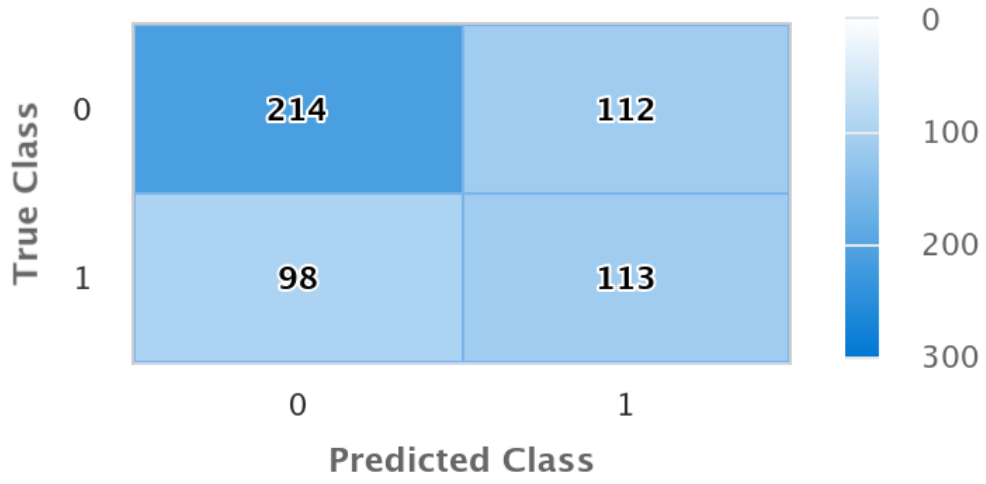
Cohorts	Sample size	Accuracy score	F1 score	False positive rate	False negative rate	Selection rat
race in African-American	783	0.663	0.623	0.193	0.468	0.37
race in Caucasian	537	0.609	0.518	0.344	0.464	0.419

Confusion matrices:

African Americans:



Caucasians:



Then let's do the same with different genders

[30]:
```
train_data, test_data = train_test_split(compas_data, test_size=0.25,␣
 ↪random_state=42, stratify=compas_data['two_year_recid'])
feature_columns = train_data.columns[:-1]

train_data['sex'] = train_data['sex'].apply(lambda x:1 if x == 'Female' else 0)
train_data['race'] = train_data['race'].apply(lambda x:1 if x == 'Caucasian'␣
 ↪else 0)
train_data['c_charge_degree'] = train_data['c_charge_degree'].apply(lambda x:1␣
 ↪if x == 'F' else 0)
train_data['Age_category'] = age_category_encoder.
 ↪fit_transform(train_data['Age_category'])

privileged_groups = [{'sex': 1}]  # Female
unprivileged_groups = [{'sex': 0}]  # Males

dataset = BinaryLabelDataset(
    favorable_label=0,
    unfavorable_label=1,
    df=train_data,
    label_names=['two_year_recid'],
```

```
    protected_attribute_names=['sex'],
    unprivileged_protected_attributes=[0]
)

reweighing = Reweighing(unprivileged_groups=unprivileged_groups,␣
 ↪privileged_groups=privileged_groups)
compas_data_reweight = reweighing.fit_transform(dataset)

train_data = pd.DataFrame(data=compas_data_reweight.features, columns=dataset.
 ↪feature_names)
train_data['two_year_recid'] = compas_data_reweight.labels.ravel()

train_data['sex'] = train_data['sex'].map({0.0: 'Male', 1.0: 'Female'})
train_data['race'] = train_data['race'].map({1: 'Caucasian', 0:␣
 ↪'African-American'})
train_data['c_charge_degree'] = train_data['c_charge_degree'].map({1.0: 'F', 0.
 ↪0: 'M'})
train_data['Age_category'] = age_category_encoder.
 ↪inverse_transform(train_data['Age_category'].astype(int))

X_train, y_train = split_label(train_data, target_feature)
X_test, y_test = split_label(test_data, target_feature)
pipeline = create_classification_pipeline(X_train)

y_train = y_train[target_feature].to_numpy()
y_test = y_test[target_feature].to_numpy()

model = pipeline.fit(X_train, y_train,␣
 ↪model__sample_weight=compas_data_reweight.instance_weights)

##RAI dashboard
feature_metadata = FeatureMetadata(categorical_features=categorical_features,␣
 ↪dropped_features=[])

rai_insights = RAIInsights(model, train_data, test_data, target_feature,␣
 ↪'classification',
                           feature_metadata=feature_metadata)

rai_insights.explainer.add()
rai_insights.error_analysis.add()

rai_insights.compute()

ResponsibleAIDashboard(rai_insights, cohort_list=cohort_list)
```

```
================================================================================
Causal Effects
```

```
Current Status: Generating Causal Effects.
Current Status: Finished generating causal effects.
Time taken: 0.0 min 0.00022499996703118086 sec
================================================================================
================================================================================
Counterfactual
Time taken: 0.0 min 1.0299962013959885e-05 sec
================================================================================

`sparse` was renamed to `sparse_output` in version 1.2 and will be removed in
1.4. `sparse_output` is ignored unless you leave `sparse` to its default value.

================================================================================
Error Analysis
Current Status: Generating error analysis reports.
Current Status: Finished generating error analysis reports.
Time taken: 0.0 min 0.11126209993381053 sec
================================================================================
================================================================================
Explanations
Current Status: Explaining 8 features
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of
testing was 0.000139 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 70
[LightGBM] [Info] Number of data points in the train set: 3958, number of used
features: 8
[LightGBM] [Info] Start training from score -0.099494

categorical_feature keyword has been found in `params` and will be ignored.
Please use categorical_feature argument of the Dataset constructor to pass this
parameter.

Current Status: Explained 8 features.
Time taken: 0.0 min 0.36919330002274364 sec
================================================================================
ResponsibleAI started at http://localhost:8711
```

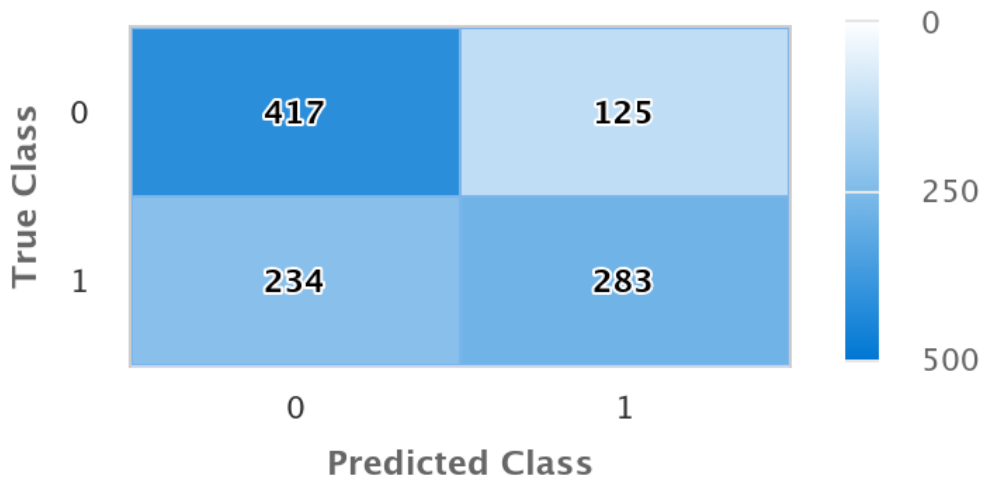
[30]: <raiwidgets.responsibleai_dashboard.ResponsibleAIDashboard at 0x219f20cb010>

Results for genders after reweighing

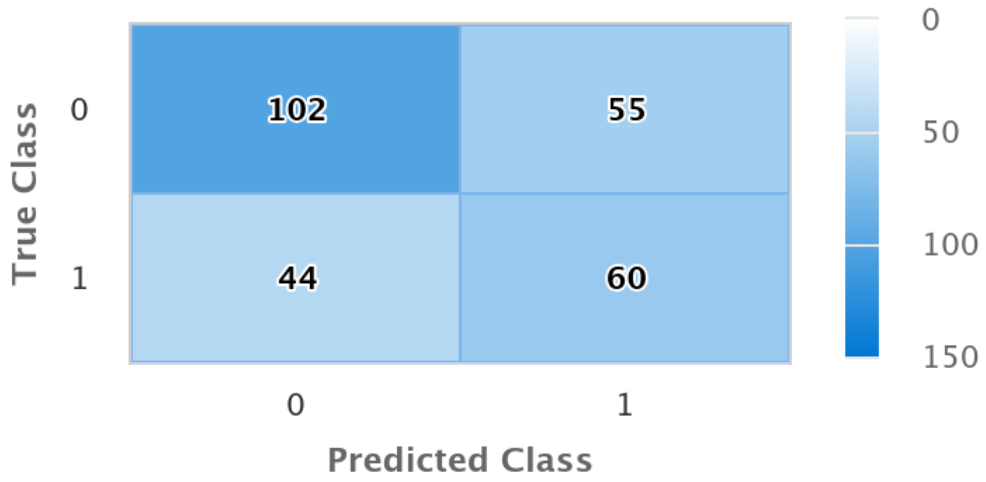
Sex metrics

Cohorts	Sample size	Accuracy sc...	False positiv...	False negati...	Selection rate
sex in Male	1 059	0.661	0.231	0.453	0.385
sex in Female	261	0.621	0.35	0.423	0.441

Males:



Females:


True Class
0
1
102
55
44
60
0
1
Predicted Class
0
50
100
150