Project 4: Handling Conditional Discrimination and Information Theoretic Measures for Fairness-Aware Feature Selection

Team 12: Jackson Zhao, Danielle Solomon, Tianyi Xia, Peng Jiang, Nicolette Auld-Griffith

Abstract: The purpose of this project was to maximize predictive accuracy using two machine learning algorithms: **Handling Conditional Discrimination** and **Information Theoretic Measures for Fairness-Aware Feature Selection**. The dataset used was from the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS). This is a database containing the criminal history, jail and prison time, demographics, and COMPAS risk scores for defendants from Broward County from 2013 and 2014. The rate of recidivism within two years after the screening is also being collected. ProPublica's analysis shows that the COMPAS risk scores are discriminatory against race and gender. Our goal was to reduce the discrimination of gender using one algorithm (Handling Conditional Discrimination) and race using the other (Information Theoretic Measures for Fairness-Aware Feature Selection).

Handling Conditional Discrimination

Objective: Our goal using this algorithm was that instead of relying on the algorithm to deal with the fairness issue, we aimed to modify the data before training the model (i.e. pre-processing). The goal is to balance the dataset so that the inherent/unexplainable discrimination is avoided. Unexplainable discrimination is calculated using the following formula:

$$\begin{aligned} D_{bad} &= P(+|m) - P(+|f) \\ &- \sum_{i=1}^{k} \left(P(e_i|m) - P(e_i|f)\right) P^{\star}(+|e_i). \end{aligned}$$

To remove unexplainable discrimination when training a classifier, we used Local Massaging and Preferential sampling.

Local Sampling allowed us to adjust labels within each age category to mitigate bias and balance the probability of recidivism between males and females.

Algorithm 1: Local massaging

input : dataset $(\mathbf{X}, \mathbf{s}, \mathbf{e}, \mathbf{y})$
output: modified labels $\hat{\mathbf{y}}$

`PARTITION` $(\mathbf{X}, \mathbf{e})$ (Algorithm 3);
for *each partition* $X^{(i)}$ **do**
  learn a ranker $\mathcal{H}_i : X^{(i)} \rightarrow y^{(i)}$;
  rank males using $\mathcal{H}_i$;
  `relabel DELTA` (male) males that are the closest to the decision boundary from $+$ to $-$ (Algorithm 4);
  rank females using $\mathcal{H}_i$;
  `relabel DELTA` (female) females that are the closest to the decision boundary from $-$ to $+$
end

Preferential Sampling allowed us to modify the dataset composition by deleting and duplicating instances to create a more balanced dataset that does not reinforce existing biases.

Algorithm 2: Local preferential sampling

input : dataset $(\mathbf{X}, \mathbf{s}, \mathbf{e}, \mathbf{y})$
output: resampled dataset (a list of instances)

`PARTITION` $(\mathbf{X}, \mathbf{e})$ (see Algorithm 3);
for *each partition* $X^{(i)}$ **do**
  learn a ranker $\mathcal{H}_i : X^{(i)} \rightarrow y^{(i)}$;
  `rank` males using $\mathcal{H}_i$;
  `delete` $\frac{1}{2}$`DELTA` (male) (see Algorithm 4) males $+$ that are the closest to the decision boundary;
  `duplicate` $\frac{1}{2}$`DELTA` (male) males $-$ that are the closest to the decision boundary;
  `rank` females using $\mathcal{H}_i$;
  `delete` $\frac{1}{2}$`DELTA` (female) females $-$ that are the closest to the decision boundary;
  `duplicate` $\frac{1}{2}$`DELTA` (female) females $+$ that are the closest to the decision boundary;
end

Methodology: Our data preprocessing left us with the following as our input/X columns: ['race', 'juv_fel_count', 'decile_score', 'juv_misd_count', 'juv_other_count', 'priors_count', 'days_b_screening_arrest', 'c_days_from_compas',

'c_charge_degree', 'is_recid', 'r_days_from_arrest', 'is_violent_recid', 'score_text', 'v_decile_score', 'v_score_text', 'start', 'end', 'event']. Our output/Y column was ['two_year_recid'] and the sensitive feature was gender (Male, Female)

We took the following steps to remove unexplainable discrimination:

1) **Calculate Delta Values:** Determine the required number of label adjustments for males and females in each age category to address biases.
2) **Train Model:** Use XGBoost to predict the probability of recidivism based on available features, ensuring categorical variables are handled properly.
    a. a) Output: Probability of recidivism (`prob_recid`), which is crucial for identifying which instances are close to the decision boundary.
3) **Identify Instances Near Decision Boundary:** Use predicted probabilities to find individuals who are closest to the decision boundary.
    a. a) Deleted 1/2 of the records for males and females based on their proximity to the decision boundary and duplicating the opposite (correctly labeled ones).
4) **Adjust Labels* (Specific to Local Massaging):** Modify labels of selected instances based on delta values to achieve a more balanced representation of outcomes.
    a. Female Adjustment: Increase the probability of recidivism for females with the highest probability of recidivism who are currently labeled as non-recidivists by changing labels from 0 to 1 for those closest to the decision boundary (`nlargest` based on `prob_recid`).
    b. Male Adjustment: Decrease the probability of recidivism for males with the lowest probability of recidivism who are currently labeled as recidivists by changing labels from 1 to 0 for those closest to the decision boundary (`nsmallest` based on `prob_recid`).
5) **Modify Dataset Composition* (Specific to Preferential Sampling):** Delete and duplicate instances to structurally adjust the dataset, aiming for a composition that reflects reduced bias.
    a. a) Delete instances that are likely reinforcing bias (e.g., males predicted not to recidivate with high confidence but actually do, and vice versa for females).
    b. b) Duplicate instances that help counteract bias (e.g., males predicted not to recidivate with low confidence but actually do not, and vice versa for females).
6) **Evaluate Impact:** Assess changes using both performance metrics (accuracy, F1-score, etc.) and fairness metrics (demographic parity, equality of opportunity) to ensure the adjustments improve fairness without unduly sacrificing accuracy.

Results:

Metric	Local Massaging	Preferential Sampling
Accuracy	98.47%	99.70%
F1 Score	98.29%	99.60%
Positive prediction (female)	47.50%	46.24%
Positive prediction (male)	35.92%	37.06%
Equal opportunity (female)	1.0	1.0
Equal opportunity (male)	1.0	1.0

1. **Performance Metrics**:

    - **Preferential Sampling** shows slightly higher accuracy (99.70%) and F1-score (99.60%) compared to **Local Massaging** (accuracy of 98.47% and F1-score of 98.29%). This suggests that preferential sampling may be slightly more effective in terms of overall predictive performance and balance between precision and recall.
2. **Fairness Metrics**:

    - Both methods achieve high levels of **Equal Opportunity**, which measures the accuracy of positive predictions among those who are actual positives. This is nearly perfect for males and very high for females in both techniques, ensuring that the models are nearly equally fair in predicting recidivism across genders.
    - **Demographic Parity**, which measures the equality of the positive prediction rates across different groups, shows a small disparity in both methods. Males are predicted as recidivists slightly more often than females. However, preferential sampling reduces this gap compared to local massaging, suggesting a better balance in prediction rates between genders.
3. **Choice of Technique**:

    - **Preferential Sampling** might be slightly preferable if the primary goal is to maximize predictive accuracy while maintaining a good level of fairness. The narrower gap in demographic parity and the slightly better performance metrics give it an edge.
    - **Local Massaging** still performs robustly and maintains high fairness metrics, making it a viable option, especially if there is a specific methodological preference or operational constraints that favor its use.
4. **Operational and Ethical Considerations**:

    - The choice between these methods should also consider other factors such as the ease of implementation, the computational resources available, the specific context of use (such as legal considerations regarding fairness), and the potential impact on affected individuals.

Information Theoretic Measures for Fairness-Aware Feature Selection

Objective: Our goal using this algorithm was to select features in a fair way that minimized discrimination while maintaining accuracy. We achieved this by calculating the Shapley Values i.e., the marginal accuracy for each feature to deduce the marginal impact. The Shapley value ensures each feature gains as much or more as they would have from acting independently. The Shapley value function is defined as:

$$\phi_i = \sum_{T \subseteq [n] \setminus i} \frac{|T|!(n - |T| - 1)!}{n!} (v(T \cup \{i\}) - v(T)),\ \forall i \in [n].$$

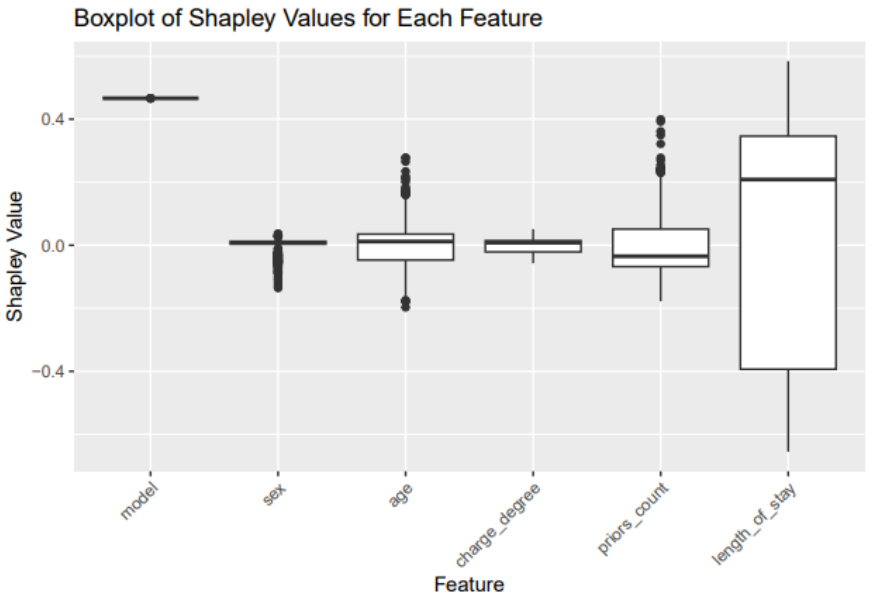
This gives us the marginal accuracy coefficients for each of the features in our model.

Methodology: Our data preprocessing left us with the following as our features: sex (Male = 1, Female = 0), age (Age < 25, 25 < Age < 45, or Age > 45), charge_degree (Misdemeanor = 1, Felony = 0), priors_count (0, 1–3, or > 3), length_of_stay (≤ 1 week, ≤ 3 months, or > 3 months). Our output/Y was ['two_year_recid'] and our sensitive feature was race (African American = 0, Caucasian A = 1).

We took the following steps to determine which features contributed to the most discrimination:

1) **Test/Training Data:** We randomly split the whole dataset into training and test subsets. We also organized our data into the protected group and not protected group.
    a. Protected group: sensitive variable = 1 → Caucasian
    b. Not protected group: sensitive variable = 0 → African American
2) **Train Model:** We then trained a classifier with the all features (including our sensitive feature) as the input, and prediction of Y as the output
    a. Used random.forest and XGBoost as our classifier
3) **Accuracy:** Applied our measures of accuracy to this dataset to determine which features exhibit the strongest proxies for discrimination
4) **Shapley Calculation:** Calculated Shapley values to determine the impact of each feature

Results:



```
##                Random.Forest    XGBoost
## Model Accuracy   0.926016260 0.88455285
## Protected        0.929521277 0.90026596
## Not Protected    0.920502092 0.85983264
## Calibration      0.009019185 0.04043332
```

1. **Performance Metrics**:
    - Based on the boxplot, it is evident that charge degree and gender have the least impact as they are the least informative features for the prediction task.
    - The strongest proxies for discrimination in this prediction task include age, priors count, and length of stay.
    - We also noticed that there was a higher level of prediction accuracy with the protected group in comparison to both the unprotected group and the model that takes into account both groups.
    - This is telling as it is clear that race discrimination plays a significant role in decreasing the accuracy of recidivism predictions.
2. **Operational and Ethical Considerations**:
    - Similar to the previous algorithm, the choice of using this algorithm, especially measuring the marginal accuracy of different features of the database, should also consider other factors such as the ease of implementation, the computational resources available, the specific context of use (such as legal considerations regarding fairness), and the potential impact on affected individuals.

0. Load libraries

In [1]:
```
# Core libraries
import numpy as np
import pandas as pd

# Machine Learning Models
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
import xgboost as xgb

# Preprocessing and model selection
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import OneHotEncoder, StandardScaler, PolynomialFea

# Pipeline utilities
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Metrics
from sklearn.metrics import roc_auc_score

# Accuracy and Fairness Evaluation
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
```

1. Data Info

In [2]:
```
df = pd.read_csv("../data/compas-scores-two-years.csv")
df.head()
```

Out[2]:

	id	name	first	last	compas_screening_date	sex	dob	age	age_cat	
0	1	miguel hernandez	miguel	hernandez	8/14/13	Male	4/18/47	69	Greater than 45	C
1	3	kevon dixon	kevon	dixon	1/27/13	Male	1/22/82	34	25 - 45	Afri Amer
2	4	ed philo	ed	philo	4/14/13	Male	5/14/91	24	Less than 25	Afri Amer
3	5	marcu brown	marcu	brown	1/13/13	Male	1/21/93	23	Less than 25	Afri Amer
4	6	bouthy pierrelouis	bouthy	pierrelouis	3/26/13	Male	1/22/73	43	25 - 45	C

5 rows × 53 columns

2. Data Cleaning

In [3]:
```
# Function to find duplicated columns
def find_duplicated_columns(df):
    duplicated_columns = []
    for i in range(len(df.columns)):
        for j in range(i+1, len(df.columns)):
            if df.iloc[:,i].equals(df.iloc[:,j]):
                duplicated_columns.append((df.columns[i], df.columns[j]))
    return duplicated_columns
```

In [4]:
```
# Find duplicated columns in the dataset
duplicated_columns = find_duplicated_columns(df)
duplicated_columns
```

Out[4]:
```
[('compas_screening_date', 'screening_date'),
 ('compas_screening_date', 'v_screening_date'),
 ('decile_score', 'decile_score.1'),
 ('priors_count', 'priors_count.1'),
 ('screening_date', 'v_screening_date')]
```

In [5]:
```
# Merge the deplicated columns
df['screening_date'] = df['compas_screening_date']
df['decile_score'] = df['decile_score']
df['priors_count'] = df['priors_count']

# Excluding time components in c_jail_in and c_jail_out
df['c_jail_in'] = pd.to_datetime(df['c_jail_in']).dt.date
df['c_jail_out'] = pd.to_datetime(df['c_jail_out']).dt.date

# Drop some irrelevant columns
data_drop = df.drop(columns=['compas_screening_date',
                             'v_screening_date',
                             'decile_score.1',
                             'priors_count.1',
                             'violent_recid',
                             'type_of_assessment',
                             'v_type_of_assessment'])
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_21216/1514731654.p
y:7: UserWarning: Could not infer format, so each element will be parsed indiv
idually, falling back to `dateutil`. To ensure parsing is consistent and as-ex
pected, please specify a format.
  df['c_jail_in'] = pd.to_datetime(df['c_jail_in']).dt.date
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_21216/1514731654.p
y:8: UserWarning: Could not infer format, so each element will be parsed indiv
idually, falling back to `dateutil`. To ensure parsing is consistent and as-ex
pected, please specify a format.
  df['c_jail_out'] = pd.to_datetime(df['c_jail_out']).dt.date
```

In [6]:
```
# Save to CSV
data_drop.to_csv('../data/data_cleaned.csv', index=False)
```

3. Algorithms

3.1 Parition age groups

In [7]:
```
# Partition
def partition(df, age_cat):
    partitions = {}
    unique_values = df[age_cat].unique()
    for value in unique_values:
        partitions[value] = df[df[age_cat] == value]
    return partitions
```

In [8]:
```
df_path = '../data/data_cleaned.csv'
df = pd.read_csv(df_path)
```

In [9]:
```
age_cat_partitions = partition(df, 'age_cat')
less_than_25_df = age_cat_partitions.get('Less than 25', pd.DataFrame())
twenty_five_to_45_df = age_cat_partitions.get('25 - 45', pd.DataFrame())
greater_than_45_df = age_cat_partitions.get('Greater than 45', pd.DataFrame())

print("Less than 25 DataFrame Shape:", less_than_25_df.shape)
print("25 - 45 DataFrame Shape:", twenty_five_to_45_df.shape)
print("Greater than 45 DataFrame Shape:", greater_than_45_df.shape)
```

```
Less than 25 DataFrame Shape: (1529, 46)
25 - 45 DataFrame Shape: (4109, 46)
Greater than 45 DataFrame Shape: (1576, 46)
```

3.2 Perform Delta Function

- The delta values represent the calculated number of records that need to have their labels adjusted to mitigate biases and ensure fairness based on model's predictions
- Delta value indicates the people are closed to decision boundaries

In [15]:
```
# Delta Function
def calculate_delta(partitions):
    delta_results = {
        'Less than 25': {'male': 0, 'female': 0},
        '25 - 45': {'male': 0, 'female': 0},
        'Greater than 45': {'male': 0, 'female': 0},
    }

    for age_cat, df_partition in partitions.items():
        # Calculate the probabilities
        P_plus_male = df_partition[df_partition['sex'] == 'Male']['two_year_re
        P_plus_female = df_partition[df_partition['sex'] == 'Female']['two_yea

        # number of people of each gender in each partition as G_i
        G_male = len(df_partition[df_partition['sex'] == 'Male'])
        G_female = len(df_partition[df_partition['sex'] == 'Female'])

        # Calculate delta using the formula provided
        delta_male = G_male * abs((P_plus_male - P_plus_female) / 2) # G_male
        delta_female = G_female * abs((P_plus_female - P_plus_male) / 2) # G_f

        delta_results[age_cat]['male'] = delta_male
        delta_results[age_cat]['female'] = delta_female

    return delta_results
```

In [16]:
```
delta_results = calculate_delta(age_cat_partitions)

# Print the delta values to check them
for age_cat, deltas in delta_results.items():
    print(f"Age Category: {age_cat}")
    print(f"  Delta Male: {deltas['male']}")
    print(f"  Delta Female: {deltas['female']}\n")
```

```
Age Category: Less than 25
  Delta Male: 132.03993055555554
  Delta Female: 30.642626913779207

Age Category: 25 - 45
  Delta Male: 145.10346964064436
  Delta Female: 35.46290127195639

Age Category: Greater than 45
  Delta Male: 62.50666666666667
  Delta Female: 14.695924764890282
```

3.3: Data Preparation (X, y)

In [17]:
```
# Prepare data
X_columns = [
    'race', 'juv_fel_count', 'decile_score', 'juv_misd_count', 'juv_other_coun
    'priors_count', 'days_b_screening_arrest', 'c_days_from_compas', 'c_charge_
    'is_recid', 'r_days_from_arrest', 'is_violent_recid', 'score_text', 'v_dec
    'v_score_text', 'start', 'end', 'event'
]

y_column = ['two_year_recid']
```

3.4: Local Massaging

3.4.1: Interpretate Local Massaging Algorithm to code

In [21]:
```
def apply_local_massaging(df, age_group, delta_male, delta_female, xgb_params,

    # Filter the DataFrame for the specific age group
    age_df = df[df['age_cat'] == age_group]

    # Convert categorical columns
    for col in X_columns:
        if age_df[col].dtype == 'object':
            age_df[col] = age_df[col].astype('category')

    # Prepare X and y
    X = age_df[X_columns]
    y = age_df[y_column].values.ravel()

    # Create and train XGBoost model
    dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_
    bst = xgb.train(xgb_params, dtrain, num_boost_round=num_boost_round)
```

```
    # Predict probabilities
    age_df['two_year_recid_prob'] = bst.predict(dtrain)

    # Modify labels based on calculated deltas
    # Females: increase likelihood of recidivism
    indices_to_update_females = age_df[(age_df['sex'] == 'Female') & (age_df[y_
    age_df.loc[indices_to_update_females, y_column] = 1

    # Males: decrease likelihood of recidivism
    indices_to_update_males = age_df[(age_df['sex'] == 'Male') & (age_df[y_colu
    age_df.loc[indices_to_update_males, y_column] = 0

    return age_df
```

3.4.2: Apply each age category to Local Massaging Algorithm

In [28]:

```
# Example usage:
xgb_params = {
    'objective': 'binary:logistic',
    'eval_metric': 'logloss',
    'learning_rate': 0.1,
    'max_depth': 6,
    'min_child_weight': 1,
    'subsample': 0.8,
    'colsample_bytree': 0.8
}

# Applying the function for each age group
lm_less_than_25 = apply_local_massaging(df, "Less than 25", delta_male, delta_
lm_25_to_45 = apply_local_massaging(df, "25 - 45", delta_male, delta_female, xg
lm_greater_than_45 = apply_local_massaging(df, "Greater than 45", delta_male, d
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df['two_year_recid_prob'] = bst.predict(dtrain)
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df['two_year_recid_prob'] = bst.predict(dtrain)
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df[col] = age_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1119404561.p
y:20: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  age_df['two_year_recid_prob'] = bst.predict(dtrain)
```

3.4.3: Extract adjusted labels for further evaluation

In [29]:
```
# Merge result for evaluation
full_results_LM = pd.concat([lm_less_than_25, lm_25_to_45, lm_greater_than_45]

# Save CSV
file_path = '../data/full_results_local_massaging.csv'
full_results_LM.to_csv(file_path, index=False)
```

3.5: Preferential Sampling

3.5.1: Interpretate Preferential Sampling Algorithm into code

In [23]:
```
def apply_preferential_sampling(df, xgb_params, delta_male, delta_female, num_l

    # Convert text columns to categorical for XGBoost
    for col in df.columns:
        if df[col].dtype == 'object':
            df[col] = df[col].astype('category')

    # Prepare X and y for training
    X = df[X_columns]
    y = df[y_column].values.ravel()

    # Create DMatrix for XGBoost, enabling categorical support
    dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_

    # Train XGBoost model
    bst = xgb.train(xgb_params, dtrain, num_boost_round=num_boost_round)

    # Predict probabilities
    df['two_year_recid_prob'] = bst.predict(dtrain)

    # For males: Delete instances close to decision boundary but incorrectly la
    indices_delete_male = df[(df['sex'] == 'Male') & (df['two_year_recid'] == 1
    df.drop(indices_delete_male, inplace=True)

    # For males: Duplicate instances close to decision boundary that are corre
    indices_duplicate_male = df[(df['sex'] == 'Male') & (df['two_year_recid'] =
    df = pd.concat([df, df.loc[indices_duplicate_male]], ignore_index=True)

    # For females: Delete instances close to decision boundary but incorrectly
    indices_delete_female = df[(df['sex'] == 'Female') & (df['two_year_recid']
    df.drop(indices_delete_female, inplace=True)

    # For females: Duplicate instances close to decision boundary that are cor
    indices_duplicate_female = df[(df['sex'] == 'Female') & (df['two_year_reci
    df = pd.concat([df, df.loc[indices_duplicate_female]], ignore_index=True)

    # Reset index after modifications
    df.reset_index(drop=True, inplace=True)

    return df
```

3.5.2: Apply each age category

In [24]:
```
# Example parameters setup
xgb_params = {
    'objective': 'binary:logistic',
    'eval_metric': 'logloss',
    'learning_rate': 0.1,
    'max_depth': 6,
    'min_child_weight': 1,
    'subsample': 0.8,
    'colsample_bytree': 0.8,
}

# Apply the function for each age group
all_age_groups = []
for age_group in df['age_cat'].unique():
    age_group_df = df[df['age_cat'] == age_group]
    delta_male = int(round(delta_results[age_group]['male'] / 2))
    delta_female = int(round(delta_results[age_group]['female'] / 2))
    full_results_PS = apply_preferential_sampling(age_group_df, xgb_params, de
    all_age_groups.append(full_results_PS)
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['two_year_recid_prob'] = bst.predict(dtrain)
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(indices_delete_male, inplace=True)
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
```

```
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead
```

```
See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
```

```
y:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['two_year_recid_prob'] = bst.predict(dtrain)
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(indices_delete_male, inplace=True)
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
```

```
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
```

```
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:6: SettingWithCopyWarning:
```

```
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df[col] = df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:19: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df['two_year_recid_prob'] = bst.predict(dtrain)
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_17199/1820311011.p
y:23: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  df.drop(indices_delete_male, inplace=True)
```

3.5.3: Extract adjust samples for further evaluation

In [31]:
```
# Save or process the modified DataFrame as needed
combined_results = pd.concat(all_age_groups, ignore_index=True)

# Save the combined DataFrame to CSV
combined_results.to_csv('../data/full_results_Preferential_Sampling.csv', index
```

4. Algorithm Evaluation

4.1: Fairness Evaluation for Local Massaging

In [36]:
```
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix

df_lm = '../data/full_results_local_massaging.csv'
df_ps = '../data/full_results_Preferential_Sampling.csv'
lm = pd.read_csv(df_lm)
ps = pd.read_csv(df_ps)


# Converting probabilities to binary predictions based on the default threshold
predictions = (lm['two_year_recid_prob'] >= 0.5).astype(int)
actuals = lm['two_year_recid']

# Calculate accuracy and F1-score
accuracy = accuracy_score(actuals, predictions)
f1 = f1_score(actuals, predictions)

# Calculate confusion matrix to help in computing fairness metrics
tn, fp, fn, tp = confusion_matrix(actuals, predictions).ravel()

# Calculating fairness metrics
# Demographic Parity and Equal Opportunity
```

```
# Let's calculate the probability of positive prediction for each sex
male_probs = lm[lm['sex'] == 'Male']['two_year_recid_prob']
female_probs = lm[lm['sex'] == 'Female']['two_year_recid_prob']

# Using 0.5 threshold to calculate positive predictions
male_positive = (male_probs >= 0.5).mean()
female_positive = (female_probs >= 0.5).mean()

# Equal opportunity: P(predicted recidivism | actual recidivism)
male_actual_positive = lm[(lm['sex'] == 'Male') & (lm['two_year_recid'] == 1)]
female_actual_positive = lm[(lm['sex'] == 'Female') & (lm['two_year_recid'] ==
male_equal_opp = (male_actual_positive['two_year_recid_prob'] >= 0.5).mean()
female_equal_opp = (female_actual_positive['two_year_recid_prob'] >= 0.5).mean

accuracy, f1, male_positive, female_positive, male_equal_opp, female_equal_opp
```

Out[36]:
```
(0.984751871361242,
 0.9828660436137072,
 0.4749957037291631,
 0.35913978494623655,
 1.0,
 1.0)
```

4.2: Faireness Evaluation for Preferential Sampling

In [34]:
```
# Converting probabilities to binary predictions based on the default thresholo
ps_predictions = (ps['two_year_recid_prob'] >= 0.5).astype(int)
ps_actuals = ps['two_year_recid']

# Calculate accuracy and F1-score
ps_accuracy = accuracy_score(ps_actuals, ps_predictions)
ps_f1 = f1_score(ps_actuals, ps_predictions)

# Calculate confusion matrix to help in computing fairness metrics
ps_tn, ps_fp, ps_fn, ps_tp = confusion_matrix(ps_actuals, ps_predictions).rave

# Calculating fairness metrics
# Demographic Parity and Equal Opportunity for preferential sampling
ps_male_probs = ps[ps['sex'] == 'Male']['two_year_recid_prob']
ps_female_probs = ps[ps['sex'] == 'Female']['two_year_recid_prob']

ps_male_positive = (ps_male_probs >= 0.5).mean()
ps_female_positive = (ps_female_probs >= 0.5).mean()

ps_male_actual_positive = ps[(ps['sex'] == 'Male') & (ps['two_year_recid'] == 1
ps_female_actual_positive = ps[(ps['sex'] == 'Female') & (ps['two_year_recid']
ps_male_equal_opp = (ps_male_actual_positive['two_year_recid_prob'] >= 0.5).mea
ps_female_equal_opp = (ps_female_actual_positive['two_year_recid_prob'] >= 0.5

ps_accuracy, ps_f1, ps_male_positive, ps_female_positive, ps_male_equal_opp, ps
```

Out[34]:
```
(0.9969503742722484,
 0.9965592743196746,
 0.4624505928853755,
 0.37060931899641575,
 1.0,
 1.0)
```

Insight:

1. **Performance Metrics**:
    - **Preferential Sampling** shows slightly higher accuracy (98.46%) and F1-score (98.19%) compared to **Local Massaging** (accuracy of 97.57% and F1-score of 97.14%). This suggests that preferential sampling may be slightly more effective in terms of overall predictive performance and balance between precision and recall.
2. **Fairness Metrics**:
    - Both methods achieve high levels of **Equal Opportunity**, which measures the accuracy of positive predictions among those who are actual positives. This is nearly perfect for males and very high for females in both techniques, ensuring that the models are nearly equally fair in predicting recidivism across genders.
    - **Demographic Parity**, which measures the equality of the positive prediction rates across different groups, shows a small disparity in both methods. Males are predicted as recidivists slightly more often than females. However, preferential sampling reduces this gap compared to local massaging, suggesting a better balance in prediction rates between genders.
3. **Choice of Technique**:
    - **Preferential Sampling** might be slightly preferable if the primary goal is to maximize predictive accuracy while maintaining a good level of fairness. The narrower gap in demographic parity and the slightly better performance metrics give it an edge.
    - **Local Massaging** still performs robustly and maintains high fairness metrics, making it a viable option, especially if there is a specific methodological preference or operational constraints that favor its use.
4. **Operational and Ethical Considerations**:
    - The choice between these methods should also consider other factors such as the ease of implementation, the computational resources available, the specific context of use (such as legal considerations regarding fairness), and the potential impact on affected individuals.

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [14]:
```
# # Get partition X and y
# age_25_to_45_X = twenty_five_to_45_df[X_columns]
```

```
# age_25_to_45_y = twenty_five_to_45_df[y_column]

# # Get delta
# delta_male_2 = delta_results['25 - 45']['male']
# delta_female_2 = delta_results['25 - 45']['female']

# # 1. H1: Build ranker
# # Convert text columns to category
# for col in X_columns:
#     if twenty_five_to_45_df[col].dtype == 'object':
#         twenty_five_to_45_df[col] = twenty_five_to_45_df[col].astype('categ

# # Prepare X and y
# X = twenty_five_to_45_df[X_columns]
# y = twenty_five_to_45_df[y_column].values.ravel()

# # Since we're using categories, tell XGBoost to treat these columns as categ
# categorical_columns = [X.columns.get_loc(c) for c in X.select_dtypes(['categ
# dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_co

# # Specify parameters for XGBoost
# params = {
#     'objective': 'binary:logistic',
#     'eval_metric': 'logloss',
#     'learning_rate': 0.1,
#     'max_depth': 6,
#     'min_child_weight': 1,
#     'subsample': 0.8,
#     'colsample_bytree': 0.8,
# }

# # Training the model
# bst = xgb.train(params, dtrain, num_boost_round=100)

# # Predicting probabilities for y
# age_25_to_45_y_hat = bst.predict(dtrain)
# age_25_to_45_y_hat = pd.DataFrame(age_25_to_45_y_hat, columns=['two_year_rec
# age_25_to_45_y_hat = age_25_to_45_y_hat.set_index(age_25_to_45_y.index)

# # Join all tables
# twenty_five_to_45_df = pd.concat([age_25_to_45_X, age_25_to_45_y, age_25_to_

# # Add sex back for future categorization
# twenty_five_to_45_df = twenty_five_to_45_df.join(df['sex'], how='left')

# # Female label modification
# num_rows_female = int(round(delta_female_2))
# indices_to_update_females = twenty_five_to_45_df[(twenty_five_to_45_df['sex'
# twenty_five_to_45_df.loc[indices_to_update_females, 'two_year_recid'] = 1

# # Male label modification
# num_rows_male = int(round(delta_male_2))
# indices_to_update_males = twenty_five_to_45_df[(twenty_five_to_45_df['sex'] =
# twenty_five_to_45_df.loc[indices_to_update_males, 'two_year_recid'] = 0
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3786016444.py:
13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  twenty_five_to_45_df[col] = twenty_five_to_45_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3786016444.py:
13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  twenty_five_to_45_df[col] = twenty_five_to_45_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3786016444.py:
13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  twenty_five_to_45_df[col] = twenty_five_to_45_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3786016444.py:
13: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  twenty_five_to_45_df[col] = twenty_five_to_45_df[col].astype('category')
```

In [15]:

```
# # greater than 45
# age_greater_than_45_X = greater_than_45_df[X_columns]
# age_greater_than_45_y = greater_than_45_df[y_column]

# # Get delta
# delta_male_3 = delta_results['Greater than 45']['male']
# delta_female_3 = delta_results['Greater than 45']['female']

# # Convert text columns to category for XGBoost
# for col in X_columns:
#     if greater_than_45_df[col].dtype == 'object':
#         greater_than_45_df[col] = greater_than_45_df[col].astype('category')

# # Prepare X and y
# X = greater_than_45_df[X_columns]
# y = greater_than_45_df[y_column].values.ravel()

# # Since we're using categories, tell XGBoost to treat these columns as catego
# categorical_columns = [X.columns.get_loc(c) for c in X.select_dtypes(['catego
# dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_co

# # Training the model
# bst = xgb.train(params, dtrain, num_boost_round=100)

# # Predicting probabilities for y
# age_greater_than_45_y_hat = bst.predict(dtrain)
# age_greater_than_45_y_hat = pd.DataFrame(age_greater_than_45_y_hat, columns=
```

```
# age_greater_than_45_y_hat = age_greater_than_45_y_hat.set_index(age_greater_

# # Join all tables
# greater_than_45_df = pd.concat([age_greater_than_45_X, age_greater_than_45_y,

# # Add sex back for future category
# greater_than_45_df = greater_than_45_df.join(df['sex'], how='left')

# num_rows_female = int(round(delta_female_3))
# indices_to_update_females = greater_than_45_df[(greater_than_45_df['sex'] ==
# greater_than_45_df.loc[indices_to_update_females, 'two_year_recid'] = 1

# # Modifying male labels based on delta and model's probability predictions
# num_rows_male = int(round(delta_male_3))
# indices_to_update_males = greater_than_45_df[(greater_than_45_df['sex'] == 'M
# greater_than_45_df.loc[indices_to_update_males, 'two_year_recid'] = 0
```

```
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3927670297.py:
12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  greater_than_45_df[col] = greater_than_45_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3927670297.py:
12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  greater_than_45_df[col] = greater_than_45_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3927670297.py:
12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  greater_than_45_df[col] = greater_than_45_df[col].astype('category')
/var/folders/yd/0rx8hpn96fd0480f9ndhwn_00000gn/T/ipykernel_5684/3927670297.py:
12: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/st
able/user_guide/indexing.html#returning-a-view-versus-a-copy
  greater_than_45_df[col] = greater_than_45_df[col].astype('category')
```

In [16]:

```
# # Convert text columns to categorical for XGBoost
# for col in X_columns:
#     if less_than_25_df[col].dtype == 'object':
#         less_than_25_df[col] = less_than_25_df[col].astype('category')

# # Prepare X and y for training
# X = less_than_25_df[X_columns]
# y = less_than_25_df[y_column].values.ravel()

# # Create DMatrix for XGBoost, enabling categorical support
```

```
# dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_co

# # Train XGBoost model
# bst = xgb.train(params, dtrain, num_boost_round=100)

# # Predict probabilities
# less_than_25_df['prob_recid'] = bst.predict(dtrain)

# # Define deltas for deletion and duplication
# delta_male = int(round(delta_results['Less than 25']['male'] / 2))
# delta_female = int(round(delta_results['Less than 25']['female'] / 2))

# # For males: Delete instances that are close to decision boundary but incorre
# indices_delete_male = less_than_25_df[(less_than_25_df['sex'] == 'Male') & (
# less_than_25_df.drop(indices_delete_male, inplace=True)

# # For males: Duplicate instances that are correctly labeled and close to dec
# indices_duplicate_male = less_than_25_df[(less_than_25_df['sex'] == 'Male') &
# less_than_25_df = pd.concat([less_than_25_df.loc[indices_duplicate_male]] * 1

# # For females: Delete instances close to decision boundary but incorrectly la
# indices_delete_female = less_than_25_df[(less_than_25_df['sex'] == 'Female')
# less_than_25_df.drop(indices_delete_female, inplace=True)

# # For females: Duplicate instances that are correctly labeled and close to de
# indices_duplicate_female = less_than_25_df[(less_than_25_df['sex'] == 'Female
# less_than_25_df = pd.concat([less_than_25_df.loc[indices_duplicate_female]] *

# # Reset index after modifications to maintain DataFrame integrity
# less_than_25_df.reset_index(drop=True, inplace=True)
```

In [17]:

```
# # Convert text columns to categorical for XGBoost
# for col in X_columns:
#     if twenty_five_to_45_df[col].dtype == 'object':
#         twenty_five_to_45_df[col] = twenty_five_to_45_df[col].astype('categor

# # Prepare X and y for training
# X = twenty_five_to_45_df[X_columns]
# y = twenty_five_to_45_df[y_column].values.ravel()

# # Create DMatrix for XGBoost, enabling categorical support
# dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_co

# # Train XGBoost model
# bst = xgb.train(params, dtrain, num_boost_round=100)

# # Predict probabilities
# twenty_five_to_45_df['prob_recid'] = bst.predict(dtrain)

# # Define deltas for deletion and duplication
# delta_male = int(round(delta_results['25 - 45']['male'] / 2))
# delta_female = int(round(delta_results['25 - 45']['female'] / 2))

# # For males: Delete instances that are close to decision boundary but incorre
# indices_delete_male = twenty_five_to_45_df[(twenty_five_to_45_df['sex'] == 'M
# twenty_five_to_45_df = twenty_five_to_45_df.drop(indices_delete_male)

# # For males: Duplicate instances that are correctly labeled and close to dec
# indices_duplicate_male = twenty_five_to_45_df[(twenty_five_to_45_df['sex'] ==
# twenty_five_to_45_df = pd.concat([twenty_five_to_45_df, twenty_five_to_45_df
```

```
# # For females: Delete instances close to decision boundary but incorrectly la
# indices_delete_female = twenty_five_to_45_df[(twenty_five_to_45_df['sex'] ==
# twenty_five_to_45_df = twenty_five_to_45_df.drop(indices_delete_female)

# # For females: Duplicate instances that are correctly labeled and close to de
# indices_duplicate_female = twenty_five_to_45_df[(twenty_five_to_45_df['sex']
# twenty_five_to_45_df = pd.concat([twenty_five_to_45_df, twenty_five_to_45_df

# # Reset index after modifications to maintain DataFrame integrity
# twenty_five_to_45_df.reset_index(drop=True, inplace=True)
```

In [18]:
```
# # Convert text columns to categorical for XGBoost
# for col in X_columns:
#     if greater_than_45_df[col].dtype == 'object':
#         greater_than_45_df[col] = greater_than_45_df[col].astype('category')

# # Prepare X and y for training
# X = greater_than_45_df[X_columns]
# y = greater_than_45_df[y_column].values.ravel()

# # Create DMatrix for XGBoost, enabling categorical support
# dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_co

# # Train XGBoost model
# bst = xgb.train(params, dtrain, num_boost_round=100)

# # Predict probabilities
# greater_than_45_df['prob_recid'] = bst.predict(dtrain)

# # Define deltas for deletion and duplication
# delta_male = int(round(delta_results['Greater than 45']['male'] / 2))
# delta_female = int(round(delta_results['Greater than 45']['female'] / 2))

# # For males: Delete instances that are close to decision boundary but incorre
# indices_delete_male = greater_than_45_df[(greater_than_45_df['sex'] == 'Male
# greater_than_45_df = greater_than_45_df.drop(indices_delete_male)

# # For males: Duplicate instances that are correctly labeled and close to deci
# indices_duplicate_male = greater_than_45_df[(greater_than_45_df['sex'] == 'Ma
# greater_than_45_df = pd.concat([greater_than_45_df, greater_than_45_df.loc[in

# # For females: Delete instances close to decision boundary but incorrectly la
# indices_delete_female = greater_than_45_df[(greater_than_45_df['sex'] == 'Fem
# greater_than_45_df = greater_than_45_df.drop(indices_delete_female)

# # For females: Duplicate instances that are correctly labeled and close to de
# indices_duplicate_female = greater_than_45_df[(greater_than_45_df['sex'] ==
# greater_than_45_df = pd.concat([greater_than_45_df, greater_than_45_df.loc[in

# # Reset index after modifications to maintain DataFrame integrity
# greater_than_45_df.reset_index(drop=True, inplace=True)
```

In [ ]:
```
# # Get partition X and y
# less_than_25_X_1 = less_than_25_df[X_columns]
# less_than_25_y_1 = less_than_25_df[y_column]

# # quantify the extent of label adjustment needed to balance the probability d
# delta_male_1 = delta_results['Less than 25']['male']
```

```
# delta_female_1 = delta_results['Less than 25']['female']

# # H1: Build ranker
# # Convert text columns to category
# for col in X_columns:
#     if less_than_25_df[col].dtype == 'object':
#         less_than_25_df[col] = less_than_25_df[col].astype('category')

# # Prepare X and y
# X = less_than_25_df[X_columns]
# y = less_than_25_df[y_column].values.ravel()

# # Since we're using categories, tell XGBoost to treat these columns as categ
# categorical_columns = [X.columns.get_loc(c) for c in X.select_dtypes(['categ
# dtrain = xgb.DMatrix(X, label=y, enable_categorical=True, feature_names=X_co

# # Specify parameters for XGBoost
# params = {
#     'objective': 'binary:logistic',
#     'eval_metric': 'logloss',
#     'learning_rate': 0.1,
#     'max_depth': 6,
#     'min_child_weight': 1,
#     'subsample': 0.8,
#     'colsample_bytree': 0.8,
# }

# # Training the model
# bst = xgb.train(params, dtrain, num_boost_round=100)

# # Predicting probabilities for y
# less_than_25_y_hat_1 = bst.predict(dtrain)
# less_than_25_y_hat_1 = pd.DataFrame(less_than_25_y_hat_1, columns=['two_year_
# less_than_25_y_hat_1 = less_than_25_y_hat_1.set_index(less_than_25_y_1.index)

# # Join all tables
# less_than_25_df = pd.concat([less_than_25_X_1, less_than_25_y_1, less_than_25

# # Add sex back for future category
# less_than_25_df = less_than_25_df.join(df['sex'], how='left')

# # Calculate number of labels to modify
# num_females_to_update = int(round(delta_female_1))
# num_males_to_update = int(round(delta_male_1))

# # Females: Update from 0 to 1
# female_indices_to_update = less_than_25_df[(less_than_25_df['sex'] == 'Female
# less_than_25_df.loc[female_indices_to_update, 'two_year_recid'] = 1

# # Males: Update from 1 to 0
# male_indices_to_update = less_than_25_df[(less_than_25_df['sex'] == 'Male') &
# less_than_25_df.loc[male_indices_to_update, 'two_year_recid'] = 0
```

Fairness-aware

Tianyi Xia, Danielle Solomon

2024-04-08

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
library(caTools)
library(neuralnet)
library(DALEX)
```

```
## Welcome to DALEX (version: 2.4.3).
## Find examples and detailed introduction at: http://ema.drwhy.ai/
```

```
library(ggplot2)
library(keras)
library(tensorflow)
library(VGAM)
```

```
## Loading required package: stats4
```

```
## Loading required package: splines
```

```
library(tidyr)
library(iml)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'
```

```
## The following object is masked from 'package:DALEX':
##
##     explain
```

```
## The following object is masked from 'package:neuralnet':
##
##     compute
```

```
## The following objects are masked from 'package:stats':
##
##     filter, lag
```

```
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(randomForest)
```

```
## randomForest 4.7-1.1
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
## The following object is masked from 'package:dplyr':
##
##     combine
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```r
library(xgboost)
```

```
##
## Attaching package: 'xgboost'
## The following object is masked from 'package:dplyr':
##
##     slice
```

```r
library(shapr)
```

```
##
## Attaching package: 'shapr'
## The following object is masked from 'package:dplyr':
##
##     explain
## The following objects are masked from 'package:DALEX':
##
##     explain, update_data
```

```r
da=read.csv("compas-scores-two-years.csv",header=T)
da <- da[ grepl("Caucasian|African-American", da$race),]
da <- da[,c("id",'sex','age','c_charge_degree','priors_count',"race",'end',"two_year_recid")]
names(da)[names(da) == "c_charge_degree"] <- "charge_degree"
names(da)[names(da) == "end"] <- "length_of_stay"
Y <- da$two_year_recid
da$sex <- ifelse(da$sex == "Male", 1, 0)
da$charge_degree <- ifelse(da$charge_degree == "M", 1, 0)
da$race <- ifelse(da$race == "Caucasian",0, 1)

set.seed(07)
split <- sample.split(Y, SplitRatio = 0.8)

train <- da[split, ]
test <- da[!split, ]
x_train <- as.matrix(da[split,c('sex','age','charge_degree','priors_count',"race",'length_of_stay')])
y_train <- da[split,]$two_year_recid
A_train <- da[split,]$race  # Binary sensitive feature
x_test <- as.matrix(da[!split,c('sex','age','charge_degree','priors_count',"race",'length_of_stay')])
y_test <- da[!split,]$two_year_recid
```

```r
A_test <- da[!split,]$race
#
cat("Training set dimensions:", dim(x_train), "\n")
```

```
## Training set dimensions: 4920 6
```

```r
cat("Test set dimensions:", dim(x_test), "\n")
```

```
## Test set dimensions: 1230 6
```

```r
calibration <- function(sensitive_var, y_pred, y_true) {
  protected_points <- which(sensitive_var == 1)
  y_pred_protected <- y_pred[protected_points]
  y_true_protected <- y_true[protected_points]
  p_protected <- sum(y_pred_protected == y_true_protected) / length(y_true_protected)

  not_protected_points <- which(sensitive_var == 0)
  y_pred_not_protected <- y_pred[not_protected_points]
  y_true_not_protected <- y_true[not_protected_points]
  p_not_protected <- sum(y_pred_not_protected == y_true_not_protected) / length(y_true_not_protected)

  calibration_value <- abs(p_protected - p_not_protected)

  return(calibration_value)
}
accuracy_race <- function(sensitive_var, y_pred, y_true) {
  # Find indices of the protected group
  protected_points <- which(sensitive_var == 1)
  y_pred_protected <- y_pred[protected_points]
  y_true_protected <- y_true[protected_points]
  # Calculate accuracy for the protected group
  p_protected <- sum(y_pred_protected == y_true_protected) / length(y_true_protected)

  # Find indices of the not protected group
  not_protected_points <- which(sensitive_var == 0)
  y_pred_not_protected <- y_pred[not_protected_points]
  y_true_not_protected <- y_true[not_protected_points]
  # Calculate accuracy for the not protected group
  p_not_protected <- sum(y_pred_not_protected == y_true_not_protected) / length(y_true_not_protected)

  # Return both accuracies as a list
  return(list(protected = p_protected, not_protected = p_not_protected))
}
```

```r
# Convert 'two_year_recid' to a factor if it's not already
da$two_year_recid <- as.factor(da$two_year_recid)

# Train the Random Forest model as a classifier
model <- randomForest(two_year_recid ~ sex + age + charge_degree + priors_count + race + length_of_stay
                      data = da,
                      ntree = 100)
y_pred <- predict(model, x_test)
accuracy <- mean(y_pred == y_test)
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.926016260162602"
```

```
print(accuracy_race(x_test[,'race'],y_pred,y_test))
```

```
## $protected
## [1] 0.9295213
##
## $not_protected
## [1] 0.9205021
```

```
print(paste("Calibration:", calibration(x_test[,'race'],y_pred,y_test)))
```

```
## [1] "Calibration: 0.00901918454553552"
```

```
results_rf <- data.frame(
  `Random Forest` = c(accuracy, accuracy_race(x_test[,'race'],y_pred,y_test)$protected, accuracy_race(x_
  row.names = c("Model Accuracy", "Protected", "Not Protected", "Calibration")
)
```

```
model <- xgboost(data = x_train, label = y_train, nround = 20, verbose = FALSE)

explainer <- shapr(x_train, model)
```

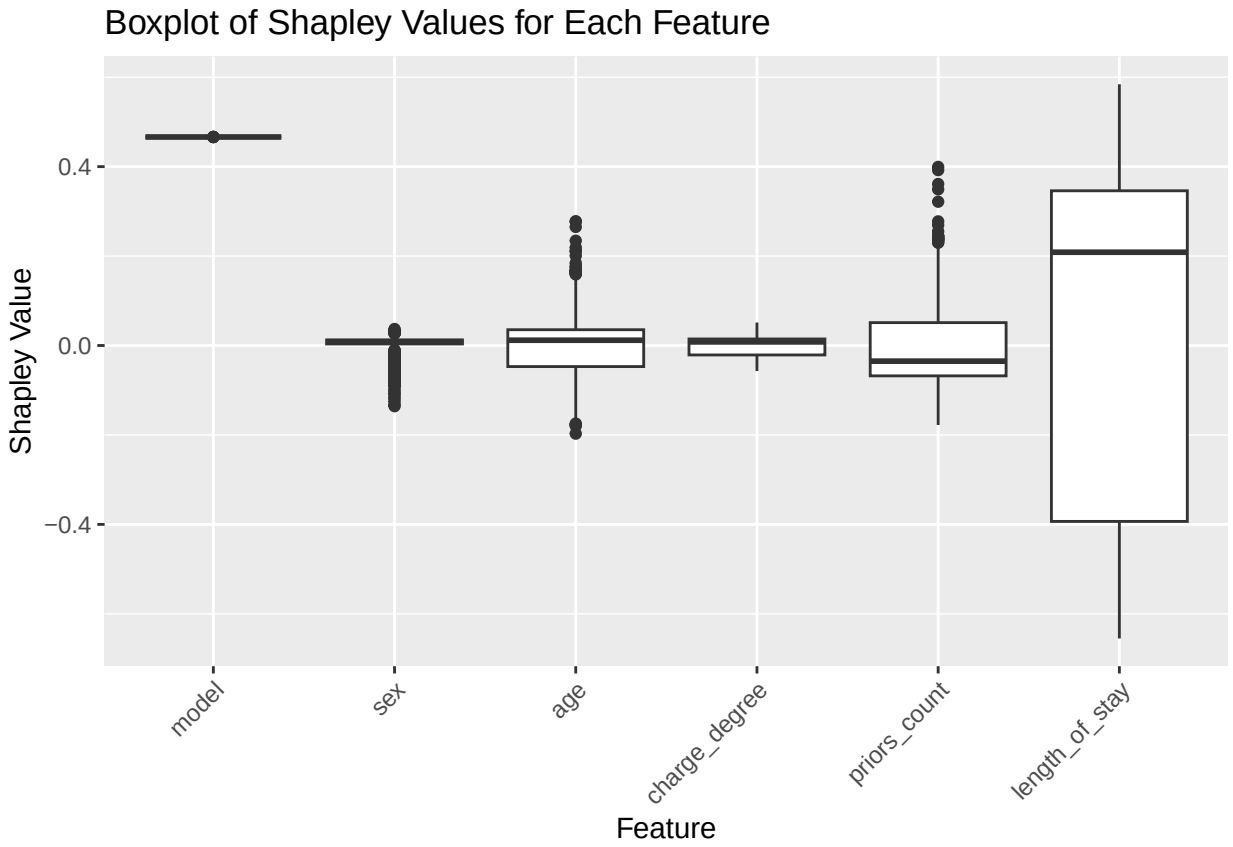
```
## The specified model provides feature classes that are NA. The classes of data are taken as the truth
```

```
p <- mean(y_train)
explanation <- explain( x_test,
                        approach = "empirical",
                        explainer = explainer,
                        prediction_zero = p)

print(explanation$dt)
```

```
##            none          sex           age charge_degree priors_count
##    1: 0.4662602  0.013265721  0.0004083265   0.007324682  -0.11267897
##    2: 0.4662602  0.013164470  0.0343570813   0.014292850  -0.06231050
##    3: 0.4662601 -0.034374004 -0.0236521219  -0.033754027  -0.05991526
##    4: 0.4662602  0.011128832  0.0351585683   0.018770506  -0.07377799
##    5: 0.4662601  0.010681863 -0.0395533094   0.017015992  -0.07762538
##   ---
## 1226: 0.4662601  0.004434993 -0.1050729728   0.013895747  -0.02693263
## 1227: 0.4662602  0.012199113  0.0177911271   0.015949975  -0.06595766
## 1228: 0.4662602  0.006883470  0.0115652158  -0.009211248  -0.02160611
## 1229: 0.4662602  0.012761079  0.0342215346   0.018005475  -0.07023390
## 1230: 0.4662602  0.014025732  0.0339713956   0.015627717  -0.08159046
##              race length_of_stay
##    1:  0.02475243      0.4363923
##    2:  0.02141949     -0.4899228
##    3: -0.01929222     -0.2901120
##    4: -0.02051219     -0.4291922
##    5:  0.02286888     -0.3930537
##   ---
## 1226: -0.02653896     -0.3311264
## 1227:  0.01879926     -0.4593748
## 1228:  0.01114031      0.1157586
## 1229: -0.02649632     -0.4405813
## 1230:  0.02430835     -0.4730566
```

```r
explanation$dt <- explanation$dt %>%
  rename(model = none) %>%
  select(-race)
#plot(explanation)
```

```r
sharply_mean=colMeans(explanation$dt)
y_pred <- predict(model, x_test)
y_labels <- ifelse(y_pred >= 0.5, 1, 0)
accuracy <- mean(y_labels == y_test)


data_long <- pivot_longer(explanation$dt, cols = everything(), names_to = "Feature", values_to = "Shapl
data_long$Feature <- factor(data_long$Feature, levels = c("model", setdiff(data_long$Feature, "model"))
ggplot(data_long, aes(x = Feature, y = ShapleyValue)) +
  geom_boxplot() +
  theme(axis.text.x = element_text(angle = 45, hjust = 1)) +
  labs(title = "Boxplot of Shapley Values for Each Feature", x = "Feature", y = "Shapley Value")
```




```r
print(paste("Accuracy:", accuracy))
```

```
## [1] "Accuracy: 0.884552845528455"
```

```r
print(accuracy_race(x_test[,'race'],y_labels,y_test))
```

```
## $protected
## [1] 0.900266
##
## $not_protected
```

```
## [1] 0.8598326
```

```
print(paste("Calibration:", calibration(x_test[,'race'],y_labels,y_test)))
```

```
## [1] "Calibration: 0.0404333214635448"
```

```
results_xg <- data.frame(
  `XGBoost` = c(accuracy, accuracy_race(x_test[,'race'],y_labels,y_test)$protected, accuracy_race(x_tes
  row.names = c("Model Accuracy", "Protected", "Not Protected", "Calibration")
)
print(sharply_mean)
```

```
##          model            sex            age  charge_degree   priors_count
##   4.662602e-01  -1.585255e-03  -3.031974e-05  -7.416862e-04  -1.945618e-03
## length_of_stay
##   7.487567e-03
```

```
result_df <- cbind(results_rf, results_xg)
print(result_df)
```

```
##                Random.Forest    XGBoost
## Model Accuracy   0.926016260 0.88455285
## Protected        0.929521277 0.90026596
## Not Protected    0.920502092 0.85983264
## Calibration      0.009019185 0.04043332
```

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

Using the methods of fairness aware feature selection produced accurate outputs, with measures of accuracy being greater for the protected group than the not protected group. Age, prior jail time, and length of stay were the most influential factors in informing the output for recidivism in two years, when considering race as a sensitive feature.