# Dependable Artificial Intelligence

*Bias Mitigation*

---

**TEAM MEMBERS:**
**Rutuja Janabandhu [B21AI017]**
**Khushi Maheshwari [B21AI052]**

## INTRODUCTION

In today's data-driven world, the use of machine learning algorithms to make critical decisions has become commonplace. However, these algorithms can inadvertently perpetuate biases present in the data they are trained on, leading to unfair or discriminatory outcomes. Detecting and mitigating bias in datasets is therefore crucial to ensure equitable and reliable results from machine learning models. In this assignment, we delve into the realm of bias detection and mitigation using a benchmark dataset. Our goal is to implement algorithms that can detect bias in the dataset and then apply techniques to mitigate this bias, ultimately verifying the effectiveness of our bias removal methods through both quantitative metrics and qualitative analysis.

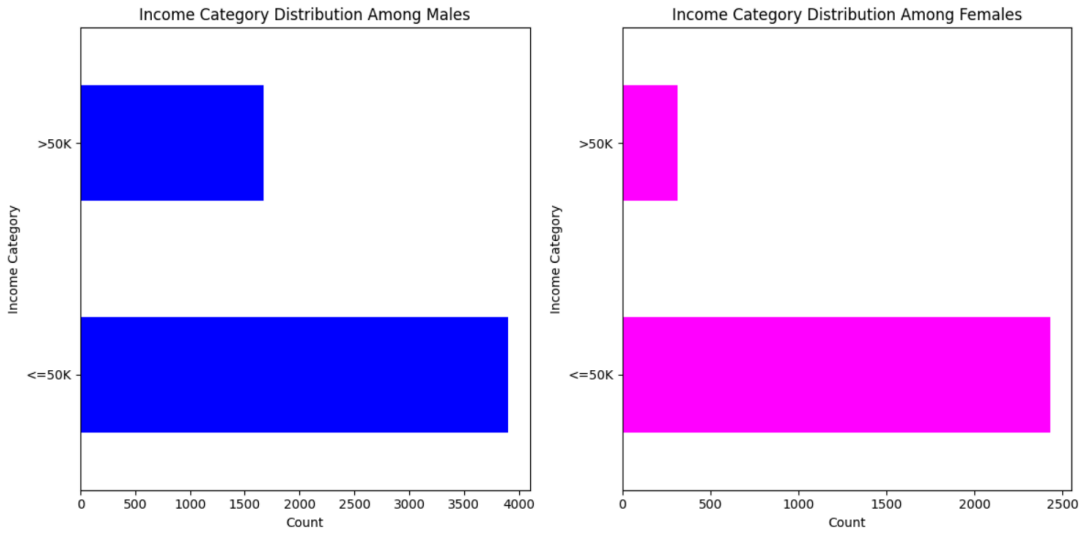
### Adult Income Dataset:

The Adult Income dataset is a widely used benchmark dataset in the field of machine learning and data science. It contains demographic information about individuals, including attributes such as age, education, occupation, and marital status, along with a binary income label indicating whether an individual earns more than $50,000 annually. This dataset is known for its inherent bias, particularly regarding factors such as race, gender, and education level, which can lead to biased predictions if not properly addressed. By working with this dataset, we aim to showcase the challenges of bias detection and mitigation in real-world datasets commonly encountered in machine learning applications.

We visualize the distribution of various features using matplotlib and seaborn.
Key observation:

- Different races have different levels of representation in terms number of datapoints with a given race
- The levels of income distribution among a given race are different
- The fraction of the population that makes less than 50K is about 3 times the fraction of the population that earns more than 50K.How would this affect the predictor trained on this data?

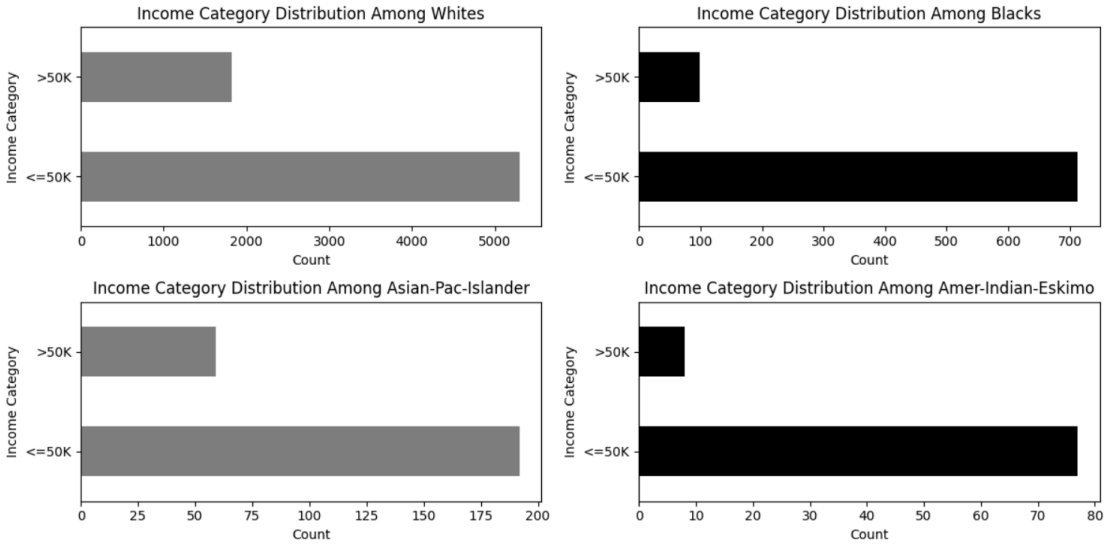
Further, we plotted two horizontal bar charts, analyzing the distribution of income categories within the male and female.




Key observation:

- The number of datapoints in the male population is considerably higher than the number of datapoints in the female category, exceeding it by more than 3 times in the higher income category.

The income category distribution among different racial demographics, specifically White, Black, Asian-Pac-Islander, and Amer-Indian-Eskimo groups, was examined using horizontal histograms.

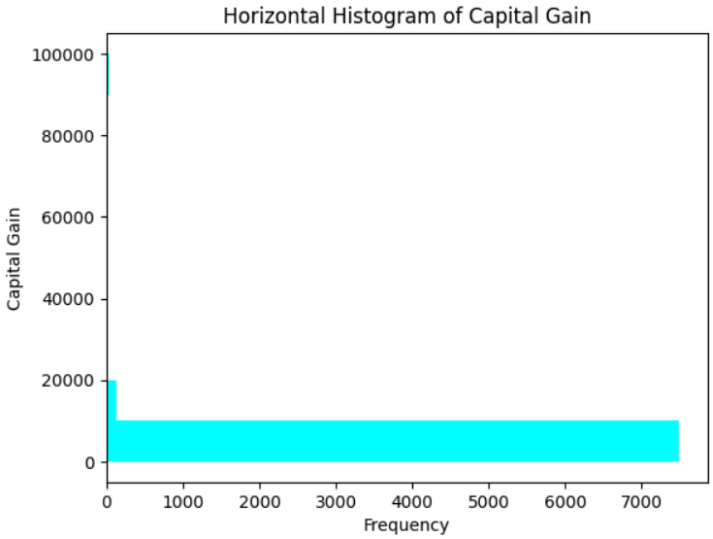


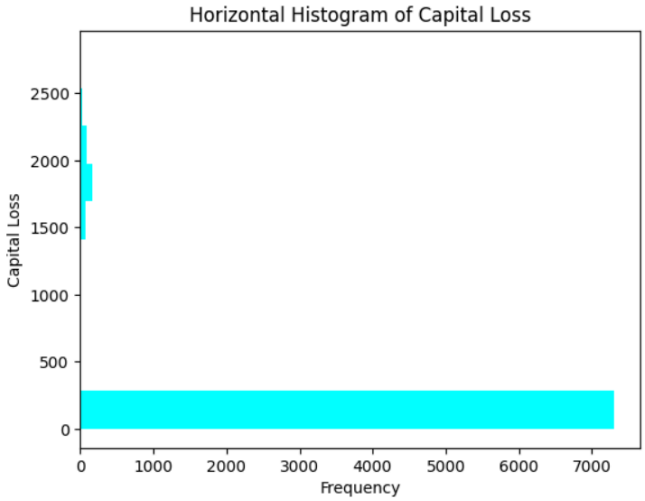

## Preprocessing

Firstly, we create a copy of the original dataset 'data' as 'datav2' to perform modifications without affecting the original data. It then identifies columns containing missing values represented as ' ?' or NaN and replaces these values with NaN using the 'replace' method and a dictionary mapping {' ?': np.nan}. Rows with any missing values are subsequently dropped from the dataset using 'dropna'.
Next, the 'native-country' column values are mapped to 'US' and 'Non-US', and labels are encoded as integers ('US_LABEL' and 'NON_US_LABEL') using a lambda function and 'map' method. Similarly, the 'salary' column values ('<=50K' and '>50K') are mapped to binary labels ('LOW_SALARY_LABEL' and 'HIGH_SALARY_LABEL'). The 'sex' column values (' Male' and ' Female') are also mapped to binary labels ('MALE_LABEL' and 'FEMALE_LABEL').
The 'marital-status' column is transformed to categorize individuals as 'Single' or 'Couple' based on predefined replacements, and the column is further encoded to integers ('COUPLE_STATUS_LABEL' and 'SINGLE_STATUS_LABEL'). The 'relationship' column is mapped to numeric values representing different relationship types.

One-hot encoding is applied to the 'relationship' and 'race' columns using OneHotEncoder from scikit-learn, and the original columns are dropped. Similarly, the 'workclass' column is transformed into categorical groups ('govt', 'private', 'self_employed', 'without_pay') using np.select() and then one-hot encoded.
The 'occupation' column is also one-hot encoded after encoding and transforming it with OneHotEncoder. Additionally, the 'capital-gain' and 'capital-loss' columns are modified to binary values based on certain conditions using np.where().

Finally, the code includes visualizations such as histograms to display the distribution of 'capital-gain' and 'capital-loss' after the modifications.




## Understanding Gender Bias in ML prediction

The evaluate_gender_performance function calculates various performance metrics across genders based on the results DataFrame. It computes overall accuracy, accuracy across gender categories, positive and negative rates, and true positive and true negative rates. These metrics provide insights into how well the model performs for different gender groups in terms of predicting salary categories.

The plot_performance_per_group function generates a scatter plot to visualize performance metrics per gender group. It plots metrics such as accuracy, positive

rates, or negative rates for each gender, providing a comparative view of model performance across genders.

The plot_comparisons_groups function is similar but allows for comparing multiple approaches or models across gender groups. It plots comparisons of performance metrics for different approaches, such as accuracy rates for male and female categories.

The plot_model_gender_metrics function is specialized for plotting model-specific gender metrics. It takes summaries of performance metrics for different models and visualizes them, allowing for a direct comparison of model performance across genders.

Finally, the model_summary function generates a summary plot for a specific model, showing key performance metrics like accuracy, positive and negative rates, true positive and true negative rates, for both male and female categories.

## Model Trained on Data with bias (without bias mitigation)

### Model Training and Evaluation

The dataset is split into training and testing sets using train_test_split from scikit-learn, with a test size of 20% and stratification based on the salary variable. The SVM (Support Vector Machine) model is then trained on the training data using the SVC class from scikit-learn. After training, the model is used to make predictions on the test set.

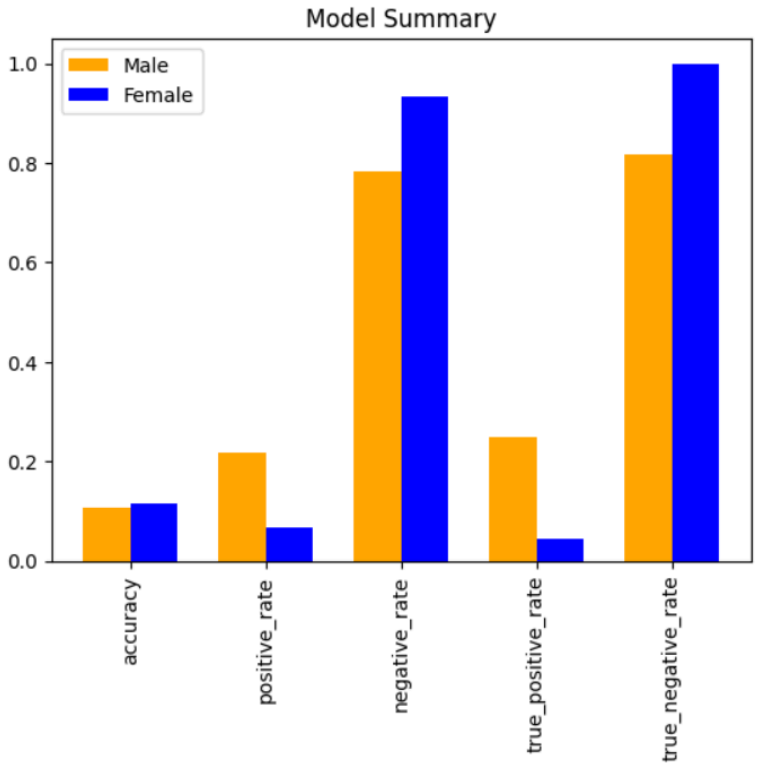
### Evaluation Metrics

The accuracy of the SVM model is evaluated using the accuracy_score function from scikit-learn. The accuracy score calculates the proportion of correctly predicted instances out of all instances in the test set. Additionally, the test_df DataFrame is created to store the test data, actual salary values, predicted salary values, and a column indicating whether each prediction is accurate or not.

```
1. Overall Accuracy: 0.05952380952380952

------------------------------------------------

2. Accuracy Across Gender:

Accuracy for Female: 0.11

Accuracy for Male: 0.11

------------------------------------------------

3. Positive Rates:

Positive rate for Female: 0.07

Positive rate for Male: 0.22

------------------------------------------------

4. Negative Rates:

Negative rate for Female: 0.93
...
True Negative Rate on Male: 0.82
```

## Evaluating Prediction Performance

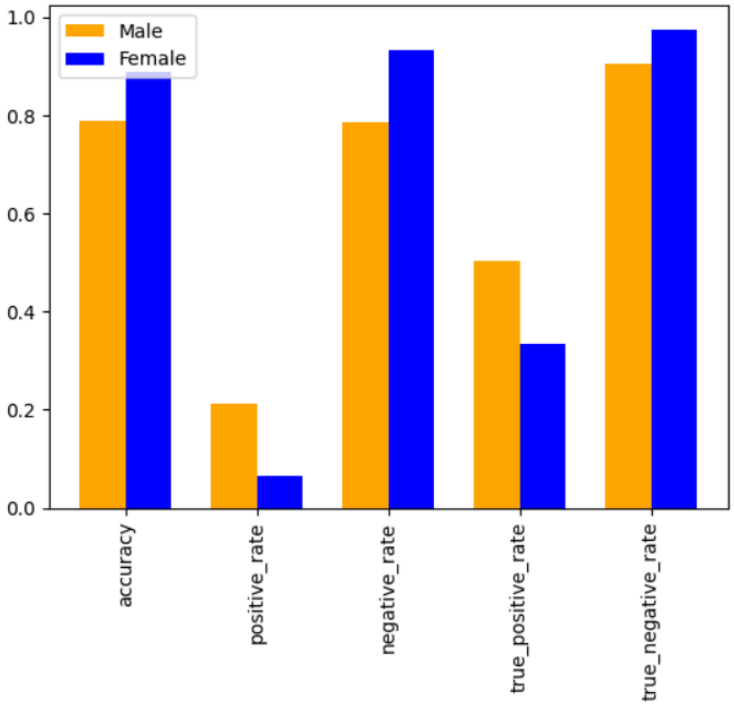
The evaluate_predictor_performance function computes summary statistics for a predictor's performance using its predictions on a test set (x_test) and corresponding labels (y_test). It essentially leverages the evaluate_gender_performance method, which calculates various performance metrics across gender categories based on the predictions and actual labels.




## Bias Mitigation Techniques

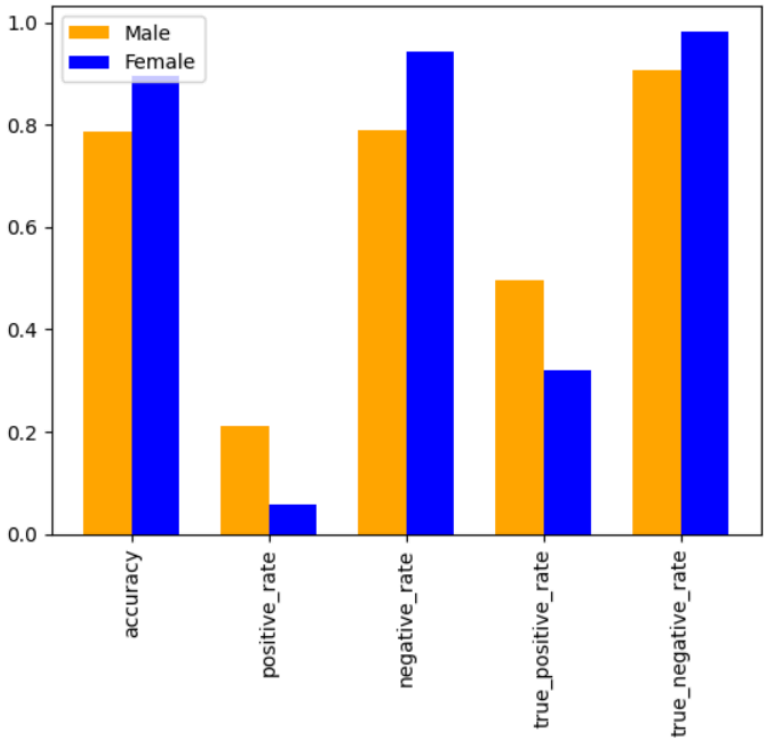
### Gender-Balanced Dataset Creation

The get_gender_balanced_dataset function creates a balanced dataset with an equal number of samples for each gender. It samples an equal number of males and females from the original dataset to ensure balanced representation in the training and testing sets. This approach is crucial for addressing biases related to gender in machine learning models.



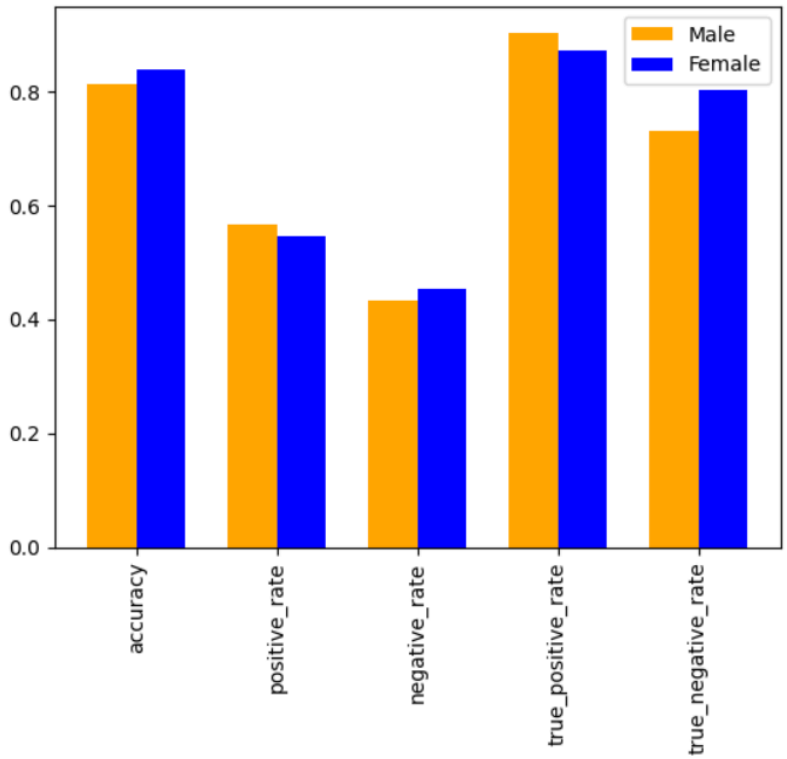

## Blind Predictor Training and Testing

The code then trains a support vector classifier (SVC) as a predictor on the gender-balanced dataset. Two approaches are considered: one where the predictor uses gender information (sex feature), and another where it operates blindly without the gender feature. This comparison helps assess whether the predictor's reliance on gender information affects its performance metrics.




## Category-Balanced Dataset Creation

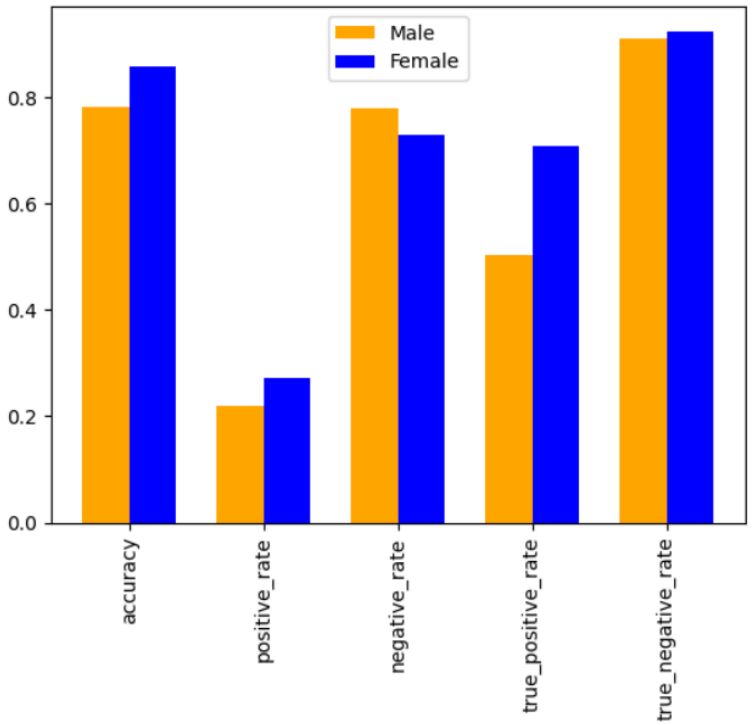
Next, the get_gender_category_balanced_dataset function creates a dataset where each gender category (male/female) has an equal number of samples across salary categories (high/low). This approach ensures balanced representation not only across genders but also across salary categories, aiding in fairness and reducing biases.



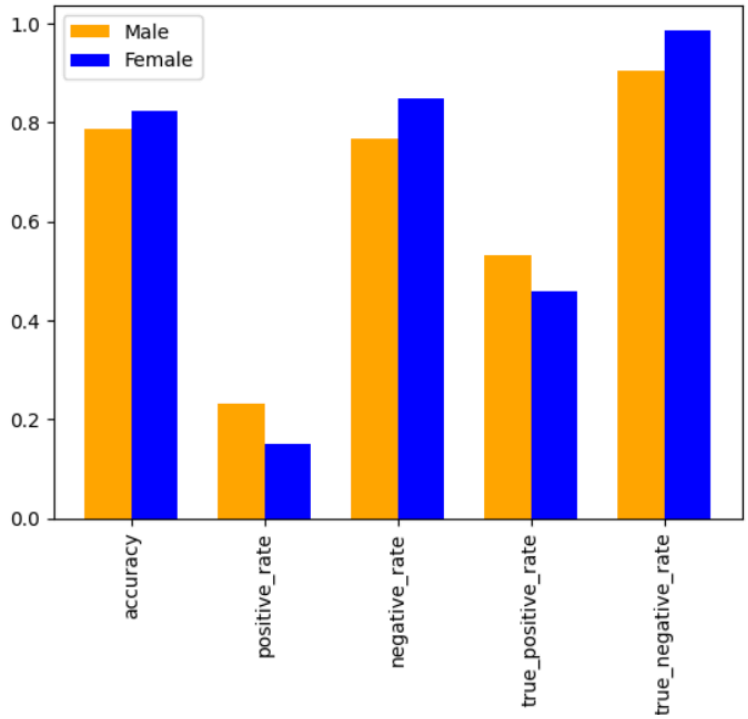

## Ratio-Balanced Dataset Creation

Lastly, the get_gender_category_ratio_balanced_dataset function creates a dataset where the ratio of samples between high and low salary categories is balanced across genders. This technique is valuable for scenarios where the distribution of high and low-income individuals differs between genders, ensuring a fair evaluation of the predictor's performance.

Equal Ratio:




Equal Ratio (Blind):



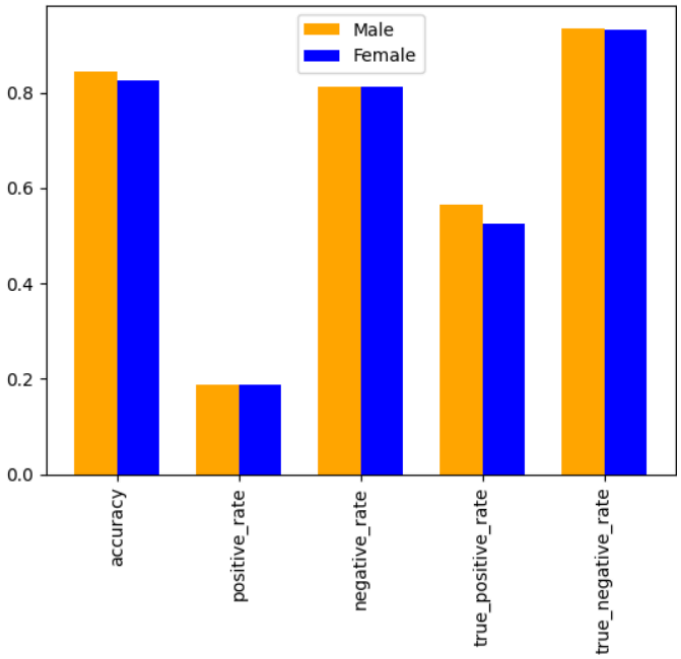
# Bias mitigation by data augmentation (generating new data)

## Model Training and Evaluation with Counterfactual Augmentation

The augmented dataset is split into training and testing sets using train_test_split from scikit-learn, with a test size of 25%. An SVC (Support Vector Classifier) model is trained on the training data with both the original and augmented features. The model is then used to predict the salary labels on the test set.

## Performance Evaluation

For the first approach, where the model is trained with the augmented dataset, predictions are made on the test data, and the performance is evaluated using the evaluate_gender_performance function. This function computes various performance metrics such as accuracy, precision, recall, and F1-score for each gender category.




## Model Summary and Comparison

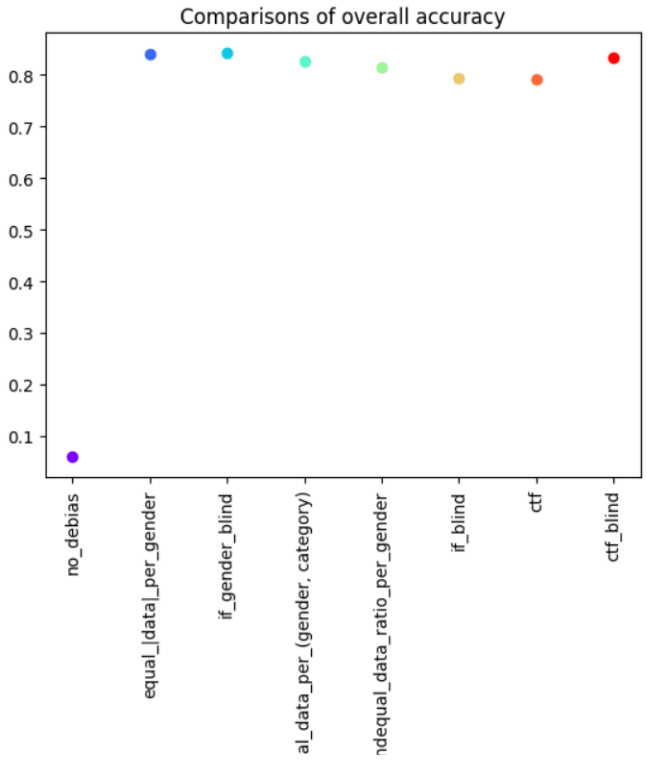
The results from the first approach are summarized and compared with the second approach, where the 'sex' feature is dropped before training the model (blind to gender). This comparison provides insights into the effectiveness of the counterfactual augmentation technique in improving model fairness and performance

across different gender categories. The model_summary function is used to summarize and compare the performance metrics between the two approaches.

## Comparing Bias Mitigation Techniques

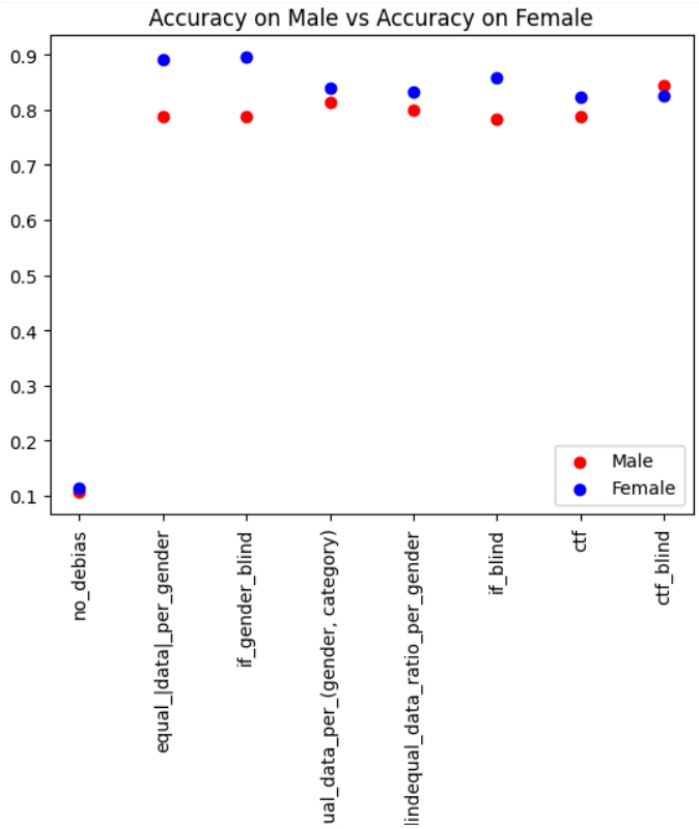
### Overall Comparisons

The plot_comparisons function is used to visualize and compare the overall accuracy across different approaches. The approaches list contains the names of the techniques being compared, while summaries contain the corresponding accuracy results for each approach. The plot helps in understanding how each approach performs in terms of overall predictive accuracy.



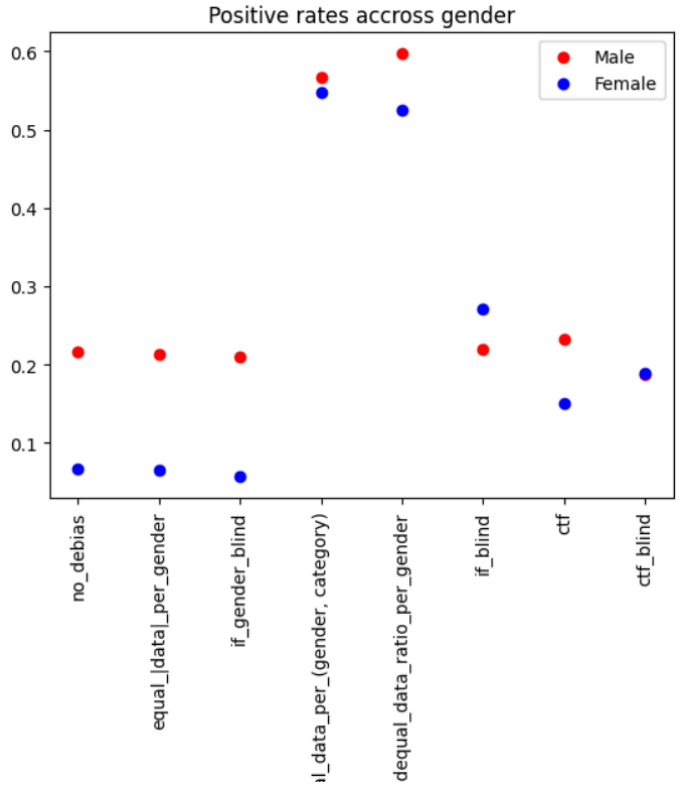

### Gender-Specific Metrics: Accuracy

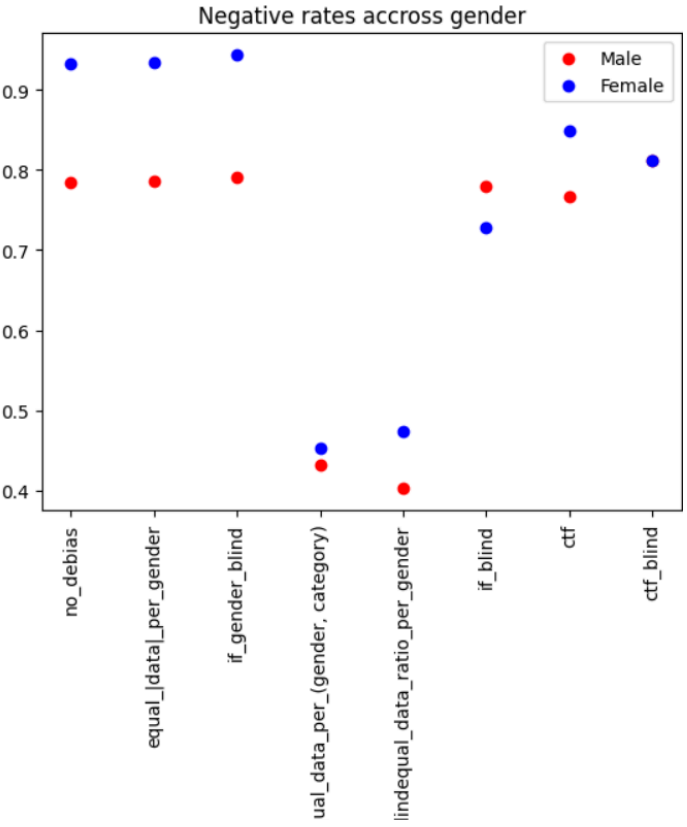
The plot_model_gender_metrics function is utilized to plot and compare accuracy metrics specifically for male and female categories across different approaches. The plots provide insights into how accurately each approach predicts outcomes for males and females, highlighting any disparities or biases in prediction performance based on gender.




## Gender-Specific Metrics: Positive and Negative Rates

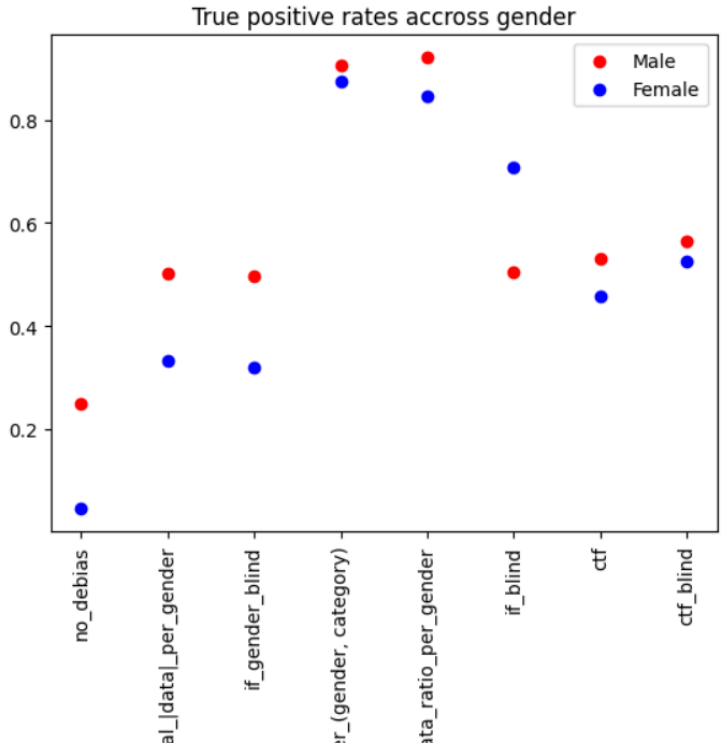
Similarly, the plot_model_gender_metrics function is used to visualize and compare positive and negative rates across gender categories for each approach. These plots shed light on how well the models handle positive and negative predictions for males and females, indicating potential biases or fairness issues in the predictions.

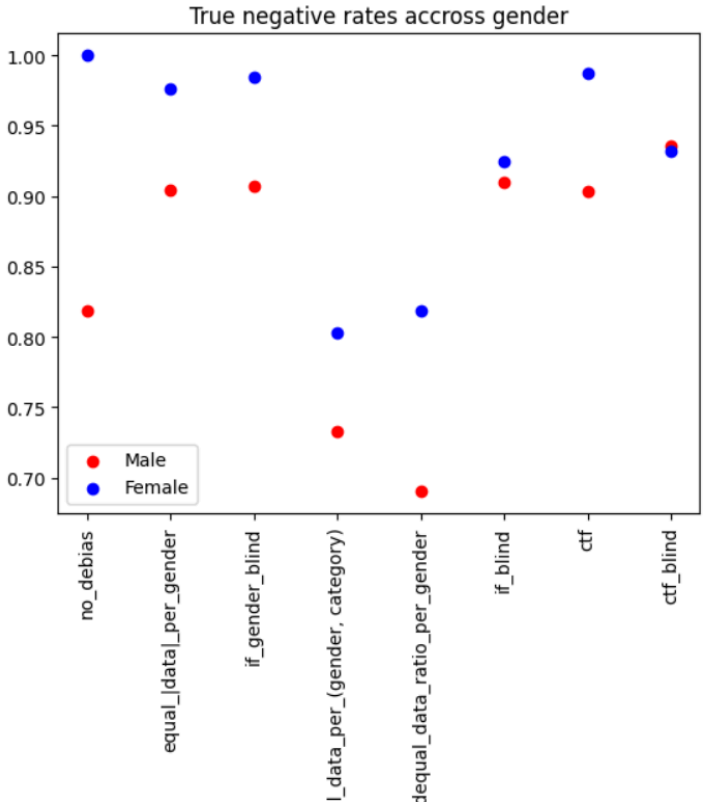

Positive rates accross gender
Male
Female
0.6
0.5
0.4
0.3
0.2
0.1
no_debias
equal_|data|_per_gender
if_gender_blind
il_data_per_(gender, category)
dequal_data_ratio_per_gender
if_blind
ctf
ctf_blind


Negative rates accross gender
Male
Female
0.9
0.8
0.7
0.6
0.5
0.4
no_debias
equal_|data|_per_gender
if_gender_blind
ual_data_per_(gender, category)
lindequal_data_ratio_per_gender
if_blind
ctf
ctf_blind

## Gender-Specific Metrics: True Positive and True Negative Rates

The last set of plots generated using plot_model_gender_metrics focuses on true positive and true negative rates across gender categories. These plots help in understanding the models' ability to correctly identify high-income and low-income individuals within each gender group, revealing any biases or discrepancies in model performance based on gender.




True negative rates accross gender
1.00
0.95
0.90
0.85
0.80
0.75
0.70
Male
Female
no_debias
equal_|data|_per_gender
if_gender_blind
l_data_per_(gender, category)
dequal_data_ratio_per_gender
if_blind
ctf
ctf_blind