Case Studies with Data: Mitigating Gender Bias on the UCI Adult Dataset

Exploring Fairness in Machine Learning

Audace Nakeshimana
Undergraduate Student & Researcher, MIT

Advised by:
Maryam Najafian
Research Scientist, MIT

Goals for this module

- In this module, we will explore steps and principles involved in building Less-Biased Machine Learning Applications.
- We look at 2 classes of techniques, specifically, data and model-based techniques for mitigating bias in Machine Learning applications.
- We will be applying these techniques on the UCI Adult Dataset, with the purpose of mitigating gender bias in predicting income category.

Module outline

1. Understanding algorithmic bias
2. Exploring University of California Irvine (UCI) adult dataset
3. Preparing data for machine learning
4. Illustrating gender bias
5. Exploring data-based debiasing techniques
6. Exploring model-based debiasing techniques
7. Conclusion

Recommended prerequisites

- Familiarity with Data Science, Statistics or Machine Learning
- Familiarity with Python, Pandas, and Scikit-Learn Library

Part 1:
Understanding Algorithmic Bias

Defining algorithmic bias, looking at its sources and implications.

Defining algorithmic/Model bias

- Throughout this module, we will use the term “bias” or “algorithmic bias” or “model bias” to describe **systematic** errors in an algorithm/model that lead to potentially **unfair** outcomes.
- We’ll qualitatively and quantitatively identify bias by looking at model **error rate disparities** across different gender demographics.
- **Note:** Throughout the module, we’ll use gender to refer biological sex at birth.
- For a more through definition of algorithmic bias, visit: https://en.wikipedia.org/w/index.php?title=Algorithmic_bias&oldid=914352968

Bias sources:
Where could algorithmic bias come from?

- Data collection

  - When data collected contains **systematic biases/stereotypes** about some demographics.

  - When there is **unequal representation** in the collected data.

- Training

  - When models are not penalized for bias.

Implications: Why is bias a problem?

Biased models/algorithms lead to:

- Unfair outcomes towards individuals/demographics.
- Further bias propagation, creating a feedback cycle of bias.

Part 2:
Exploring UCI Adult Dataset

We build familiarity with the University of California Irvine (UCI) Adult Dataset, and explore income distributions across different demographics.

UCI adult dataset overview

About Citation Policy Donate a Data Set Contact

Search

Repository Web

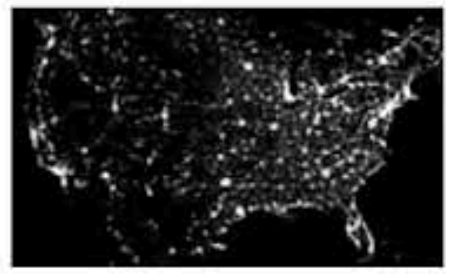
Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Adult Data Set

Download: Data Folder, Data Set Description



Abstract: Predict whether income exceeds $50K/yr based on census data. Also known as "Census Income" dataset.

Data Set Characteristics:	Multivariate	**Number of Instances:**	48842	**Area:**	Social
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	14	**Date Donated**	1996-05-01
Associated Tasks:	Classification	**Missing Values?**	Yes	**Number of Web Hits:**	1571439

Source:

Donor:

Ronny Kohavi and Barry Becker
Data Mining and Visualization
Silicon Graphics.
e-mail: ronnyk '@' live.com for questions.

Data Set Information:

Extraction was done by Barry Becker from the 1994 Census database. A set of reasonably clean records was extracted using the following conditions: ((AAGE>16) && (AGI>100) && (AFNLWGT>1)&& (HRSWK>0))

Prediction task is to determine whether a person makes over 50K a year.

source:
https://archive.ics.uci.edu/ml/datasets/Adult

MIT D-Lab USAID FROM THE AMERICAN PEOPLE CITE

UCI adult dataset overview

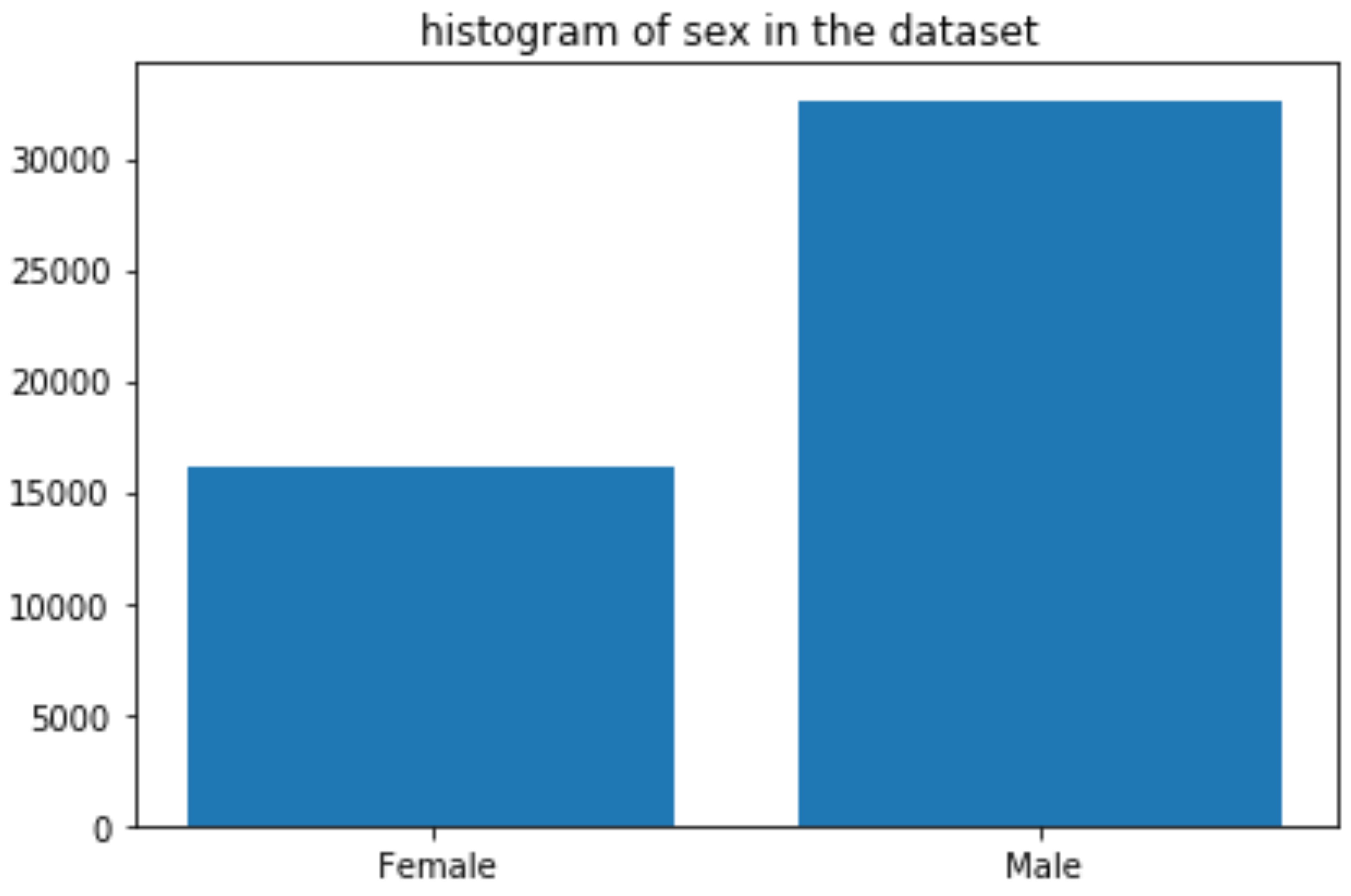
```
In [3]: data = pd.read_csv(ADULT_PATH)
        data.head()
```

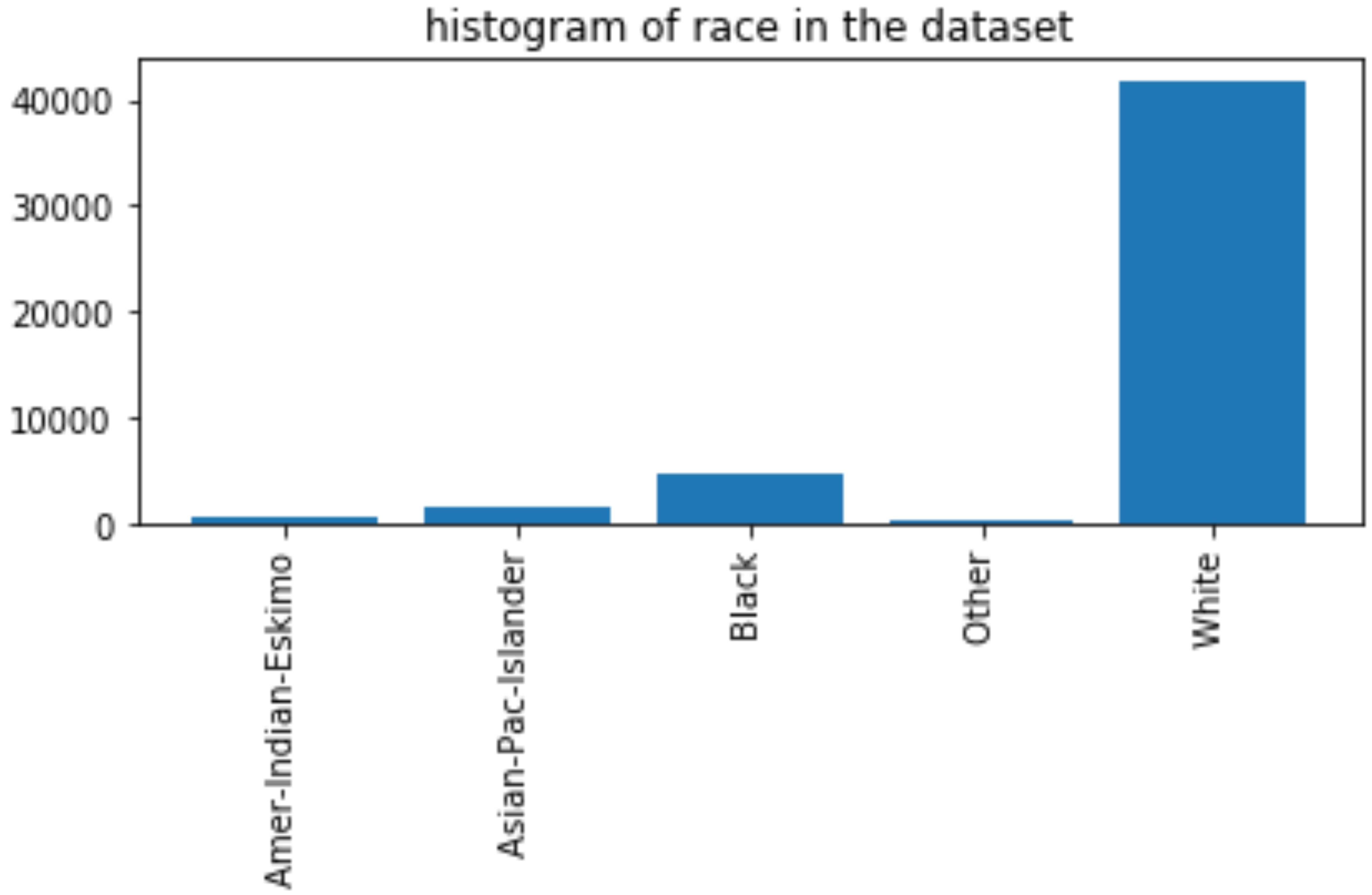
Out[3]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516.0	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	United-States	<=50K
1	50	Self-emp-not-inc	83311.0	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0	United-States	<=50K
2	38	Private	215646.0	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0	United-States	<=50K
3	53	Private	234721.0	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0	United-States	<=50K
4	28	Private	338409.0	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0	Cuba	<=50K

source:
Audace Nakeshimana & Maryam Najafian

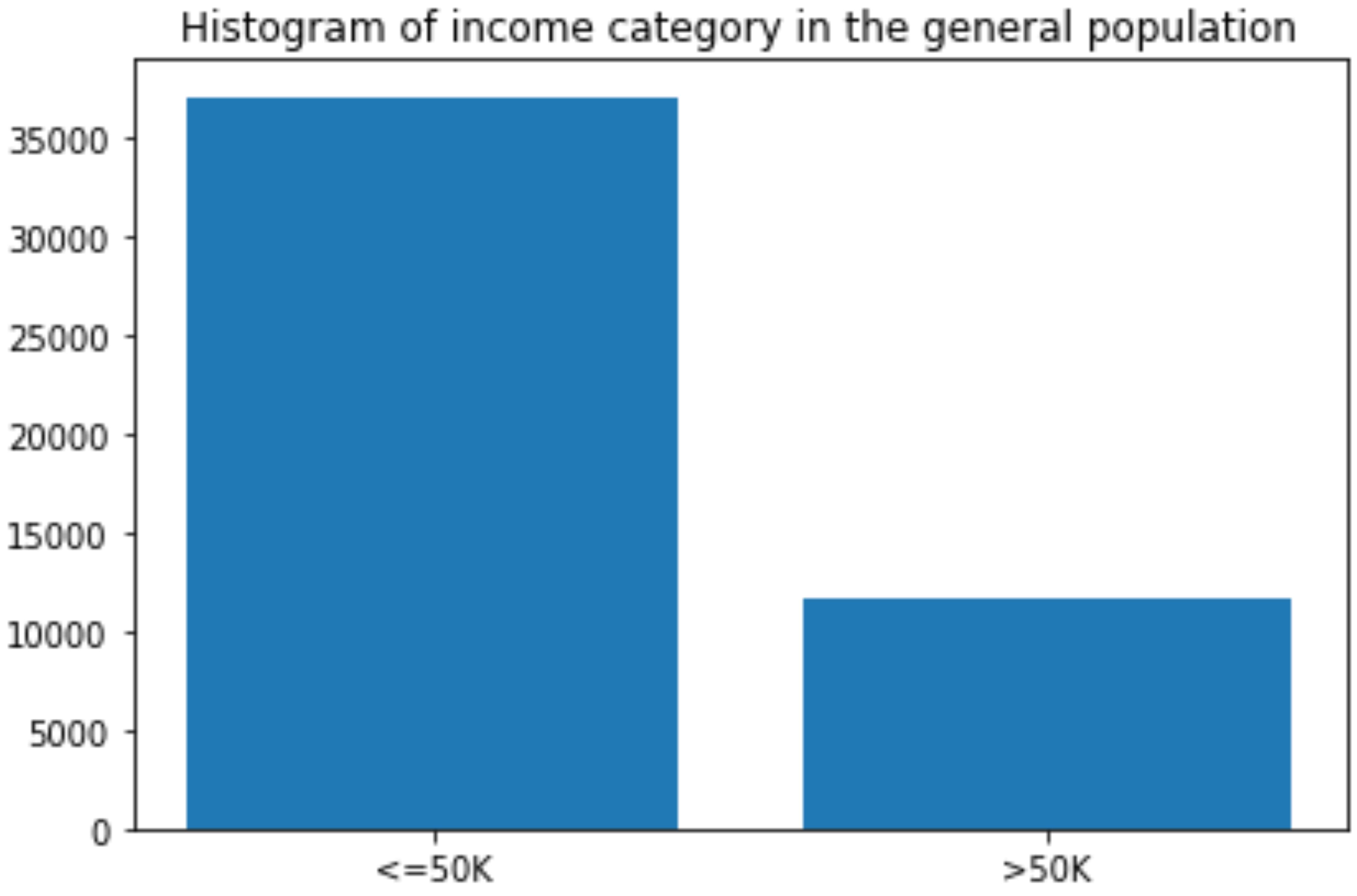
Demographic representation - Gender and race





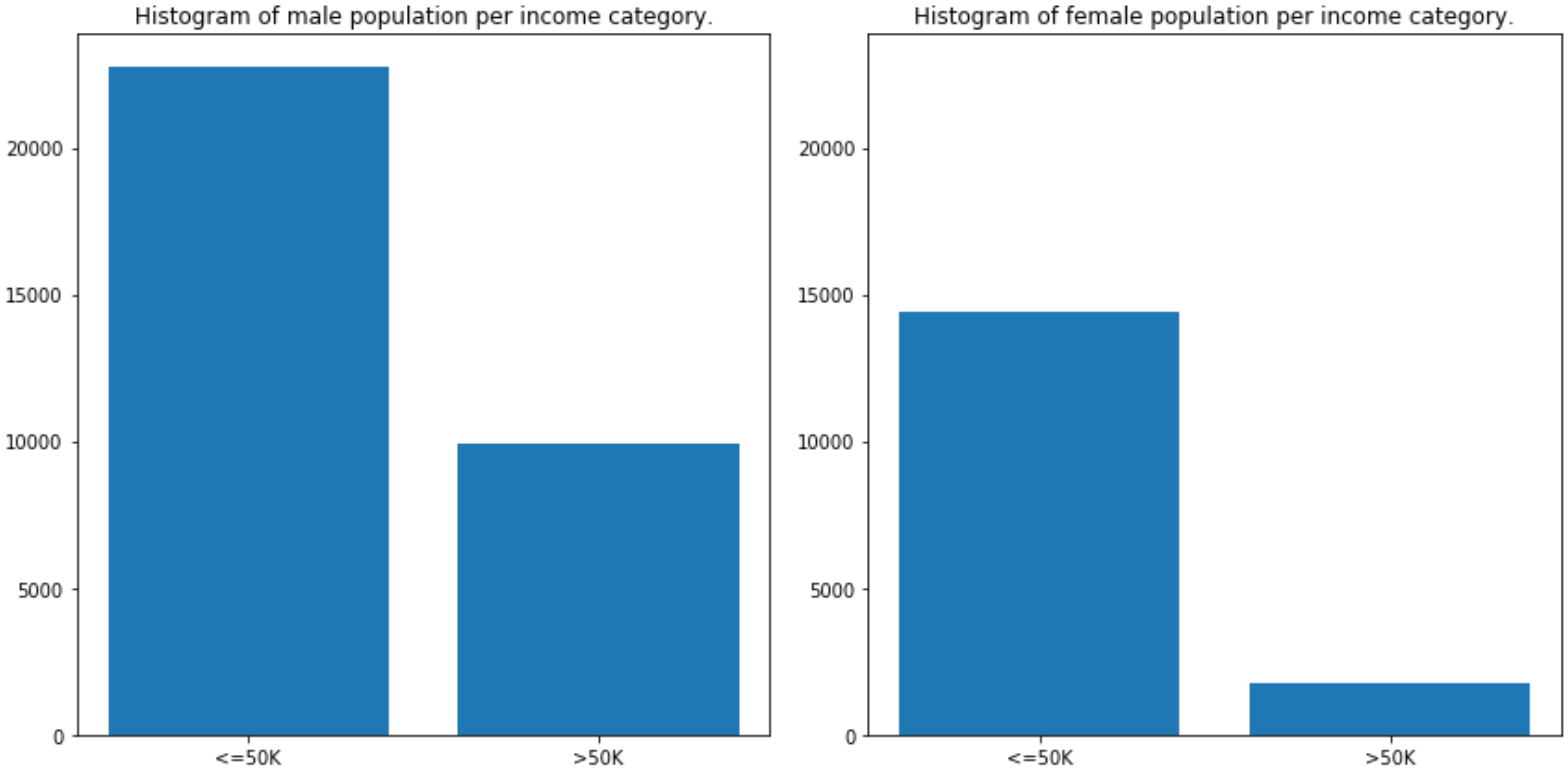
source:
Audace Nakeshimana & Maryam Najafian

Income category in the general population



source:
Audace Nakeshimana & Maryam Najafian

Income category across gender



source:
Audace Nakeshimana & Maryam Najafian

Key observations:

- 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.
- Think: How might this representation disparity affect predictions of a model trained from this data?

Part 3:
Preparing data for Machine Learning

We explore different steps involved in transforming our data from raw representation to appropriate numerical or categorical representation.

Converting native country to binary

```
In [17]: datav2[datav2['native-country'] == ' United-States'].shape
Out[17]: (41292, 15)
```

```
In [18]: datav2.loc[datav2['native-country']!=' United-States', 'native-country'] = 'Non-US'
         datav2.loc[datav2['native-country'] == ' United-States', 'native-country'] = 'US'
         US_LABEL, NON_US_LABEL = (0, 1)
         datav2['native-country'] = datav2['native-country'].map({'US':US_LABEL,'Non-US':NON_US_LABEL}).astype(int)
         datav2.head()
```

Out[18]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516.0	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	Male	2174.0	0.0	40.0	0	<=50K
1	50	Self-emp-not-inc	83311.0	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	Male	0.0	0.0	13.0	0	<=50K
2	38	Private	215646.0	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	Male	0.0	0.0	40.0	0	<=50K
3	53	Private	234721.0	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0.0	0.0	40.0	0	<=50K
4	28	Private	338409.0	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0.0	0.0	40.0	1	<=50K

source:
Audace Nakeshimana & Maryam Najafian

Converting sex and salary to binary

In [19]:
```
FEMALE_LABEL, MALE_LABEL = (0, 1)
HIGH_SALARY_LABEL, LOW_SALARY_LABEL = (0, 1)
```

In [20]:
```
datav2['salary'] = datav2['salary'].map({'>50K':HIGH_SALARY_LABEL,'<=50K':LOW_SALARY_LABEL})
datav2['sex'] = datav2['sex'].map({' Male':MALE_LABEL,' Female':FEMALE_LABEL})
datav2.head()
```

Out[20]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary
0	39	State-gov	77516.0	Bachelors	13.0	Never-married	Adm-clerical	Not-in-family	White	1	2174.0	0.0	40.0	0	1
1	50	Self-emp-not-inc	83311.0	Bachelors	13.0	Married-civ-spouse	Exec-managerial	Husband	White	1	0.0	0.0	13.0	0	1
2	38	Private	215646.0	HS-grad	9.0	Divorced	Handlers-cleaners	Not-in-family	White	1	0.0	0.0	40.0	0	1
3	53	Private	234721.0	11th	7.0	Married-civ-spouse	Handlers-cleaners	Husband	Black	1	0.0	0.0	40.0	0	1
4	28	Private	338409.0	Bachelors	13.0	Married-civ-spouse	Prof-specialty	Wife	Black	0	0.0	0.0	40.0	1	1

source:
Audace Nakeshimana & Maryam Najafian

Convert relationship to one-hot

In [24]:
```
# First convert relationship to integers
rel_map = {' Unmarried':0,' Wife':1,' Husband':2,' Not-in-family':3,' Own-child':4,' Other-relative':5}
datav2['relationship'] = datav2['relationship'].map(rel_map)
datav2.head(10)
```

In [25]:
```
# Now convert relationship from integer to one-hot
datav2 = pd.get_dummies(datav2, columns=['relationship'])
datav2.head()
```

Out[25]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	race	sex	capital-gain	capital-loss	hours-per-week	native-country	salary	relationship_0	relationship_1
0	39	State-gov	77516.0	Bachelors	13.0	1	Adm-clerical	White	1	2174.0	0.0	40.0	0	1	0	0
1	50	Self-emp-not-inc	83311.0	Bachelors	13.0	0	Exec-managerial	White	1	0.0	0.0	13.0	0	1	0	0
2	38	Private	215646.0	HS-grad	9.0	1	Handlers-cleaners	White	1	0.0	0.0	40.0	0	1	0	0
3	53	Private	234721.0	11th	7.0	0	Handlers-cleaners	Black	1	0.0	0.0	40.0	0	1	0	0
4	28	Private	338409.0	Bachelors	13.0	0	Prof-specialty	Black	0	0.0	0.0	40.0	1	1	0	1

source:
Audace Nakeshimana & Maryam Najafian

Transformations of other categorical attributes

- Other categorical attributes including relationship, race, work class, occupation, and capital-gain and capital-loss are also transformed to binary/one-hot.
- In most cases, we chose **binary encoding for simplicity**, but this is often a decision that has to be made on case by case basis.
- Converting features like work class to binary might be **problematic** if individuals from different categories have **systematically different levels of income.** However, on the other hand, not doing this might be a problem if one category has very few people that we **can’t generalize** from it.

Part 4:
Illustrating Gender Bias

We apply the standard ML approach to our data, then illustrate gender bias in performing the task of predicting income category.

Predicting income category with the standard ML approach

- scikit-learn example:

```
In [38]: (x_train, y_train), (x_test, y_test) = get_naive_dataset(datav2)
         model = MLPClassifier(max_iter=MLP_MAX_ITER)
         model.fit(x_train,y_train)
         prediction = model.predict(x_test)
```

- In the example above, we applied the standard Machine Learning approach:
  - **Split dataset** into training and test data.
  - **Select model** (MLPClassifier in this case)
  - **Fit model** on training data
  - Use model to **make predictions** on test data.

source:
Audace Nakeshimana & Maryam Najafian

Notes on MLP classifier:

- Belongs to the class of feedforward neural networks.
- Each node uses a non-linear activation function, giving it ability to separate non-linear data.
- Is trained using backpropagation technique
- Suffers overfitting and is not easily interpretable

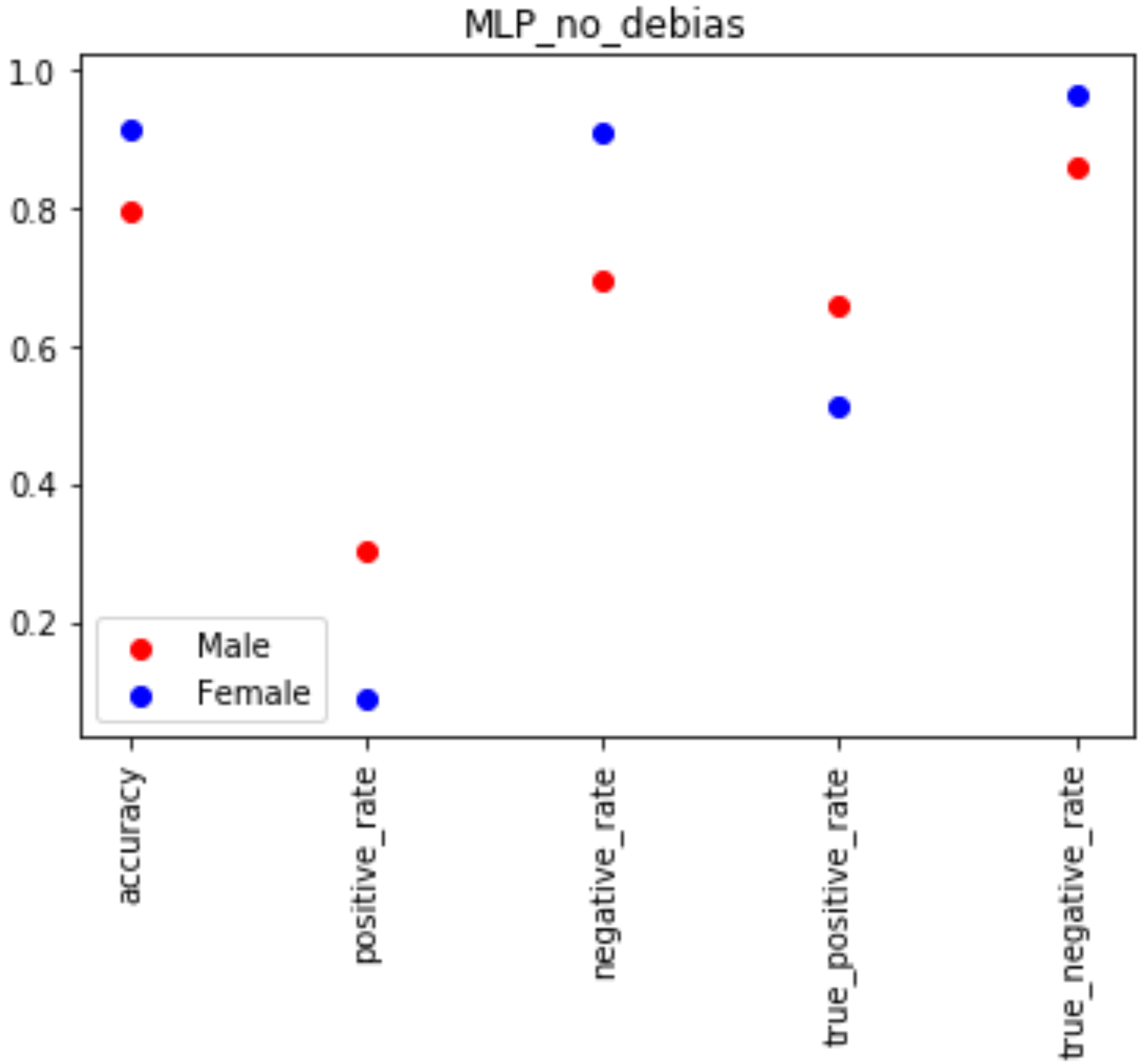
For more details, visit https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html

Evaluating error rate across gender

Let's start by establishing terminology: Throughout the rest of the module,

- **Positive** category will refer to **High Income** category (> $50K/Year)
- **Negative** category will refer to **Low Income** category (<= $50K/Year)

Evaluating error rate across gender



source:
Audace Nakeshimana & Maryam Najafian

Gender bias as error rate disparity across gender

- The metrics that we saw indicate consistent disparity in error rate between male & female.
- This is what we will define as **gender bias.**
- Mitigating gender bias is equivalent to using different techniques to **minimize this disparity**, and this will be the focus of the rest of the module.

Part 5:
Exploring Data-Based Debiasing Techniques

We explore different ways to recalibrate and augment our dataset in a way that makes predictions less biased.

Motivation

- We hypothesized that gender bias could come from unequal representation of male and female demographics.
- We therefore attempt to **re-calibrate** and **augment** the dataset with the aim to "equalize" gender representation in our training data

5.1: Debiasing by unawareness

We mitigate gender bias by removing gender from the attributes we train on.

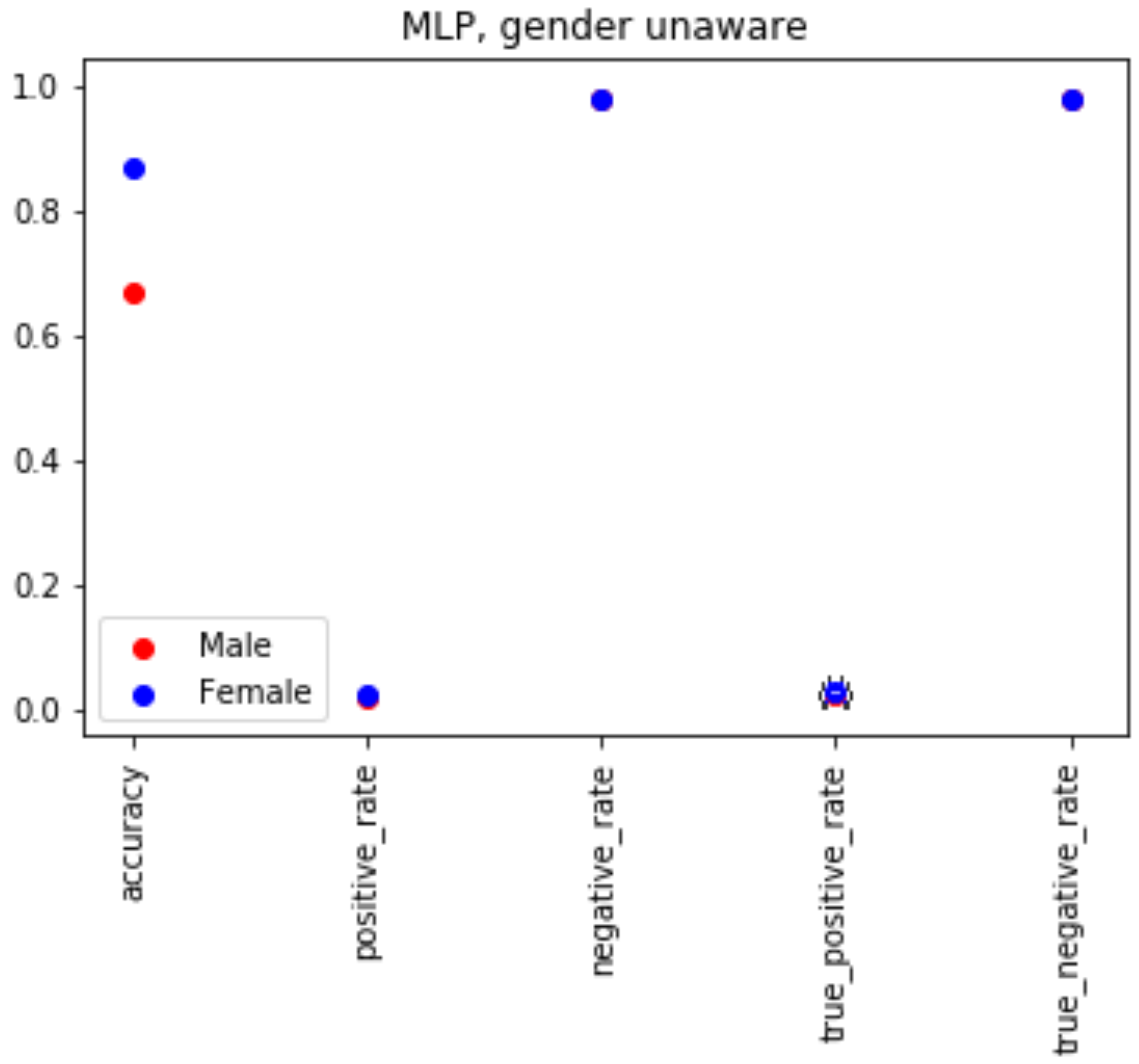
```
In [45]: def get_unawareness_dataset(dataset):
             (x_train, y_train), (x_test, y_test) = get_naive_dataset(dataset)
             testdata = x_test.copy()
             assert "sex" in list(testdata.columns), ("columns: ", list(testdata.columns))

             x_train, x_test = [v.drop(['sex'], axis=1) for v in (x_train, x_test)]
             return (x_train, y_train), (x_test, y_test), testdata
```

```
In [46]: predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
         (x_train, ytrain), (x_test, y_test), testdata = get_unawareness_dataset(datav2)
         predictor.fit(x_train, y_train)
```

source:
Audace Nakeshimana & Maryam Najafian

5.1: Debiasing by unawareness

Our run yielded the following results:



source:
Audace Nakeshimana & Maryam Najafian

MITD-Lab USAID FROM THE AMERICAN PEOPLE CITE

Comments on unawareness

- Debiasing by unawareness can be one approach to mitigate bias to some extent.
- However, studies have shown that this method can be ineffective, especially if there are other features in the dataset that correlate with the protected attribute that we are dropping.
- These are commonly referred to as proxy variables.

5.2 Equalize the number of datapoints

We attempt different approaches to "equalize" representation by using equal number/ratio of male and female individuals in our dataset.

- # Male = # Female
- # (Male, INCOME_LEVEL) = # (Female, INCOME_LEVEL)
- # (MALE, HIGH_INCOME/ #(MALE,LOW_INCOME) = #(FEMALE, HIGH_INCOME)/ #(FEMALE, LOW_INCOME)

Equal number of datapoints per gender category

Train on equal number of datapoints from the male and female demographics.

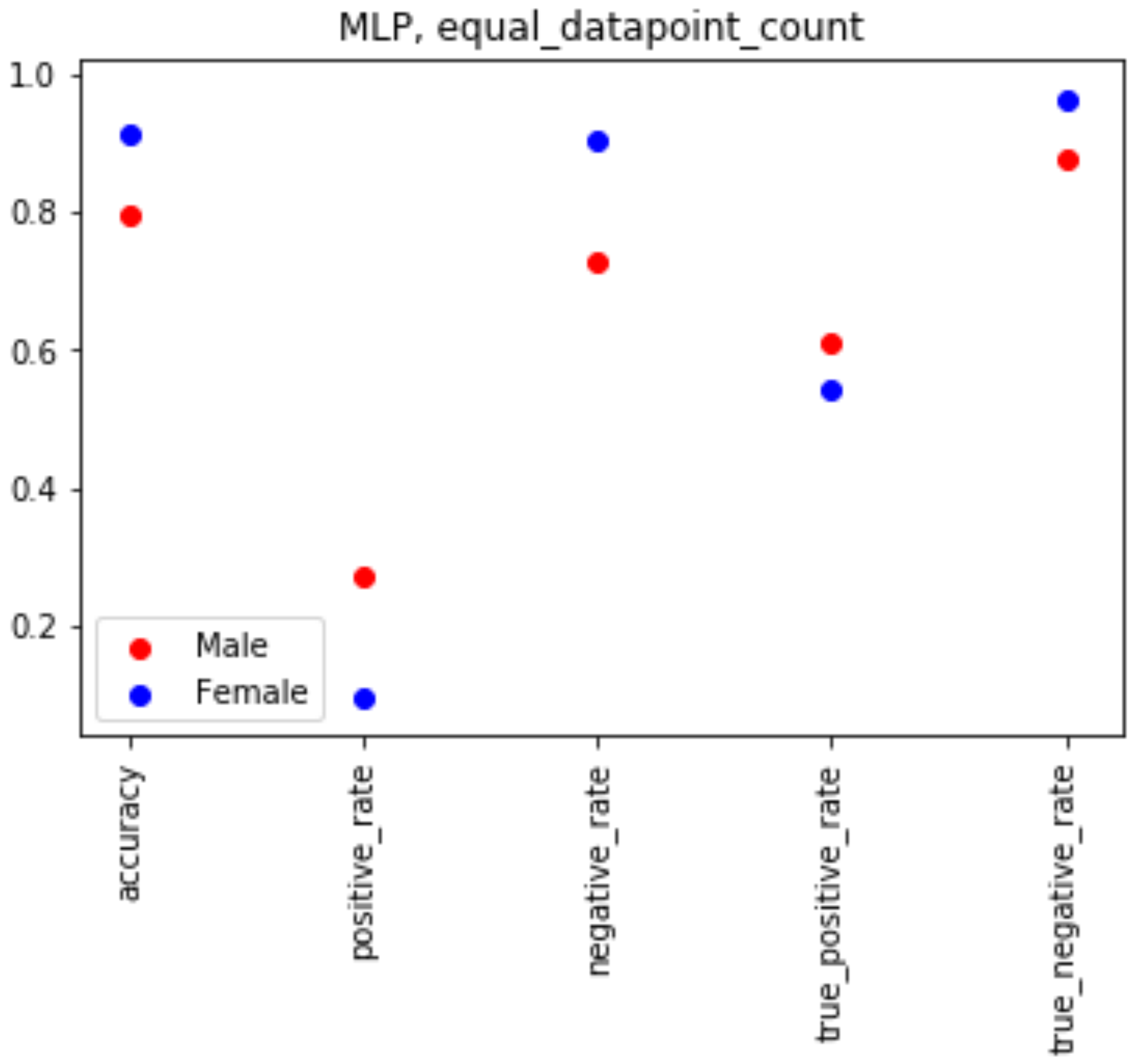
In [172]:
```
def get_gender_balanced_dataset(dataset, test_size=0.25):
    """
    Returns (x_train, y_train), (x_test, y_test) with equal number of samples for each gender
    """
    males, females = dataset[dataset.sex == MALE_LABEL], dataset[dataset.sex==FEMALE_LABEL]
    sampled_males = males.sample(n=int(min(females.shape[0], males.shape[0]))).reset_index(drop=True)
    combined = pd.concat([sampled_males, females]).sample(frac=1).reset_index(drop=True)
    Xvals=combined.drop(["salary"], axis=1)
    Yvals = combined["salary"]
    x_train, x_test, y_train, y_test = train_test_split(Xvals, Yvals, test_size=test_size)
    return (x_train, y_train), (x_test, y_test)
```

In [175]:
```
(x_train, y_train), (x_test, y_test) = get_gender_balanced_dataset(datav3)
predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
predictor.fit(x_train, y_train)
approach_2 = evaluate_predictor_performance(predictor.predict(x_test), x_test, y_test)
model_summary("MLP, equal_datapoint_count", "", approach_2)
```

source:
Audace Nakeshimana & Maryam Najafian

Equal number of datapoints per gender category

Our run yielded the following results:



source:
Audace Nakeshimana & Maryam Najafian

Equal number of datapoints per income level in each gender category

Results in a dataset in which the number of high income and low income earners is the same in each gender category.

```
In [176]: (x_train, y_train), (x_test, y_test) = get_gender_category_balanced_dataset(datav3)
          predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
          predictor.fit(x_train, y_train)
          predictions = predictor.predict(x_test)
```

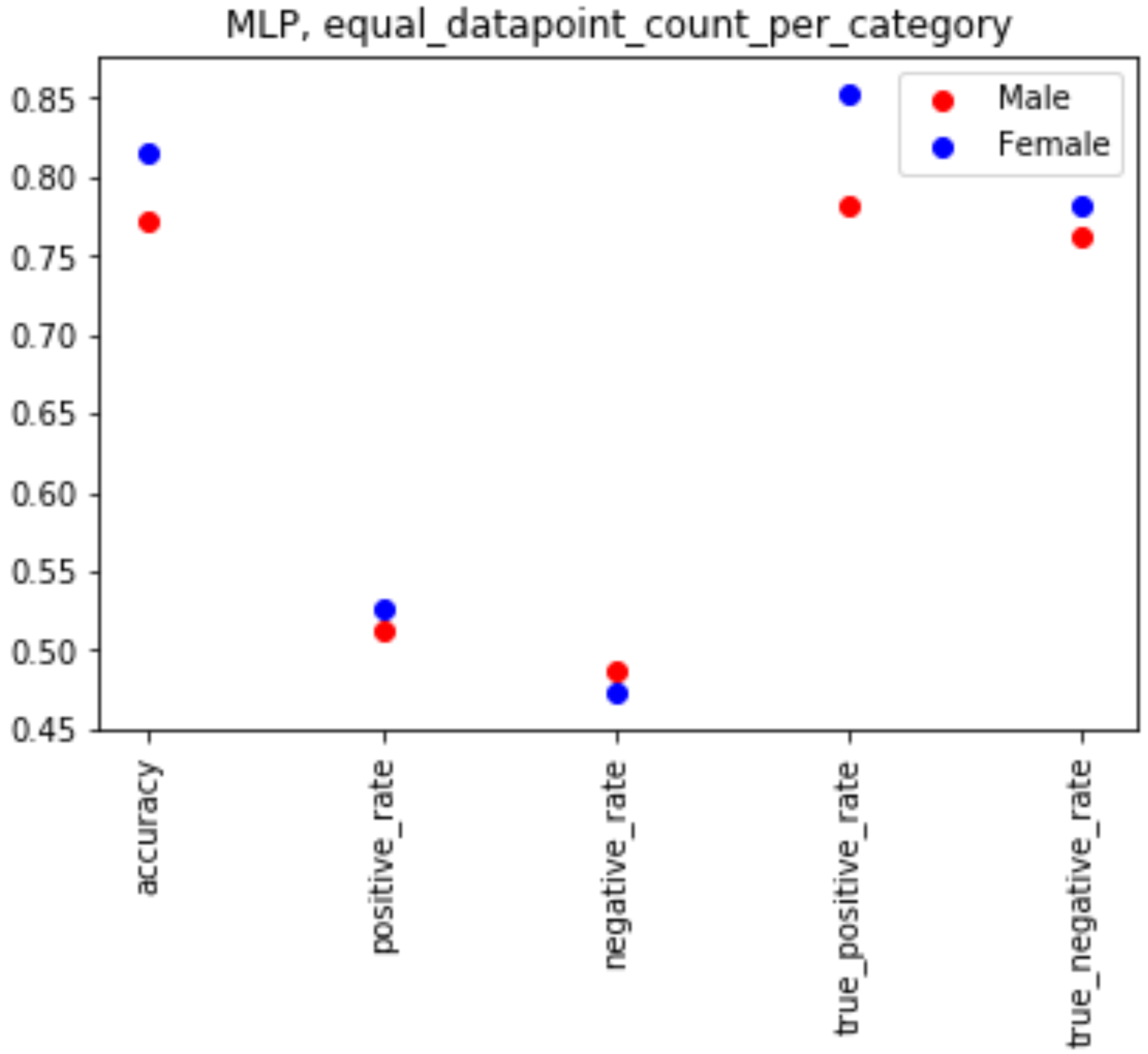
```
In [177]: approach_3 = evaluate_predictor_performance(predictions, x_test, y_test)
          model_summary("MLP, equal_datapoint_count_per_category", "", approach_3)
```

source:
Audace Nakeshimana & Maryam Najafian

MITD-Lab USAID FROM THE AMERICAN PEOPLE CITE

Equal number of datapoints per income level in each gender category

Our run yielded the following results:



source:
Audace Nakeshimana & Maryam Najafian

MITD-Lab USAID FROM THE AMERICAN PEOPLE CITE

Notes on equalizing the number of datapoints

- You might have noticed that making a selection of the dataset that equalizes the number of datapoints per demographic/category. This makes the resulting dataset size constrained to product of the size of the smallest demographic and the number of demographics.
- In some cases, equalizing the ratio instead can lead to a higher resulting sample size.

Equal ratio of the number of datapoints per income level in each category

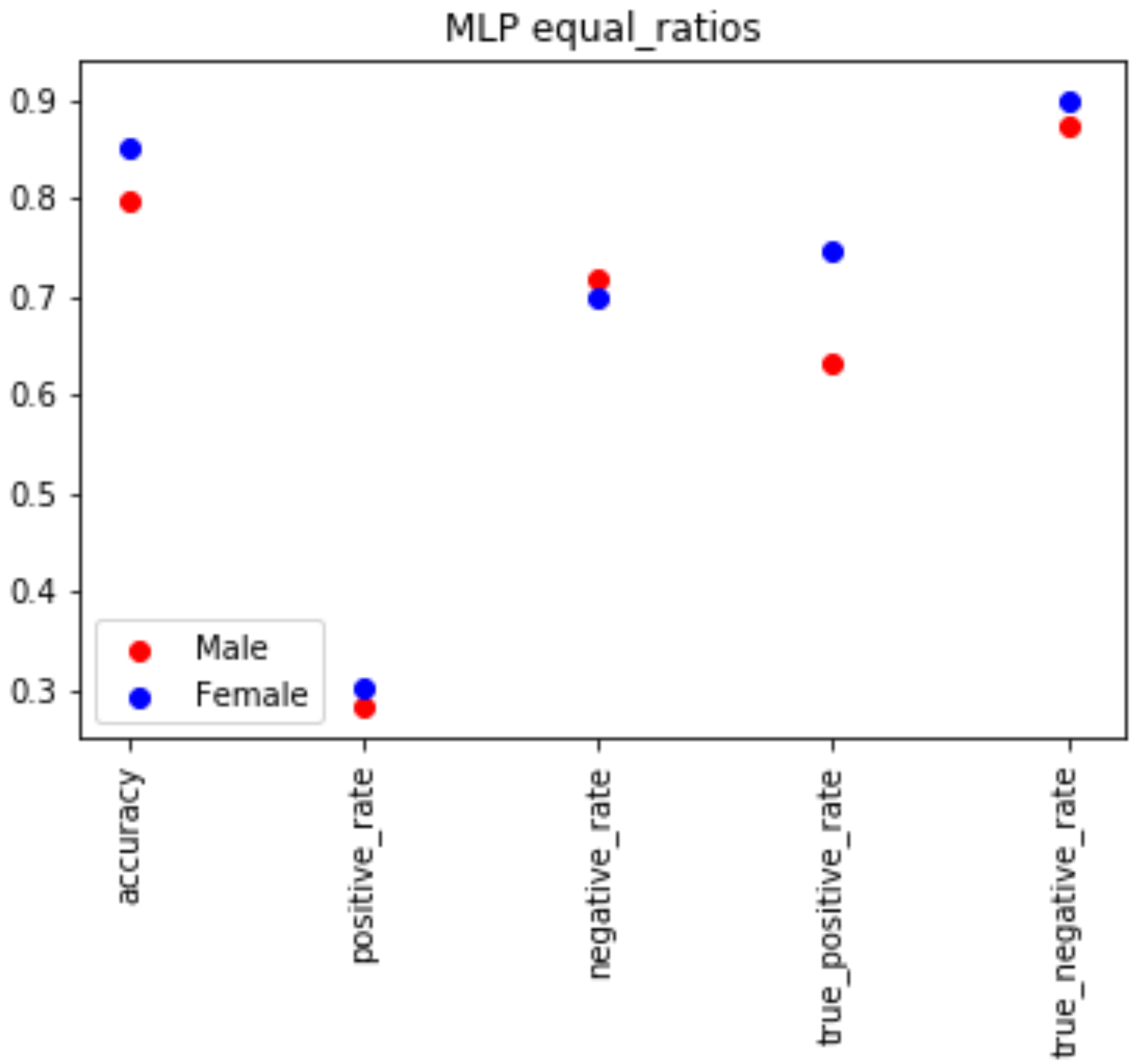
Let's equalize the ratio **of male individuals with high income to male individuals with low income** and the ratio of **female individuals with high income and female individuals with low income.** This results into a higher sample size than equalizing the number of datapoints.

```
In [62]: (x_train, y_train), (x_test, y_test) = get_gender_category_ratio_balanced_dataset(datav3)
         predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
         predictor.fit(x_train, y_train)
         predictions = predictor.predict(x_test)
```

source:
Audace Nakeshimana & Maryam Najafian

Equal ratio of the number of datapoints per income level in each category

Our run yielded the following results:



source:
Audace Nakeshimana & Maryam Najafian

5.3 Augment data with counterfactuals

Approach:

For each datapoint Xi with a given gender, generate a new datapoint Yi that only differs with Xi at the gender attribute, and add Yi to our dataset.

5.3 Augment data with counterfactuals

Task: Convince yourself that the resulting dataset will satisfy all the following constraints

- # Male = # Female
- # (Male, INCOME_LEVEL) = # (Female, INCOME_LEVEL)

And as a result:

- # (MALE, HIGH_INCOME/ #(MALE,LOW_INCOME) = #(FEMALE, HIGH_INCOME)/ #(FEMALE, LOW_INCOME)

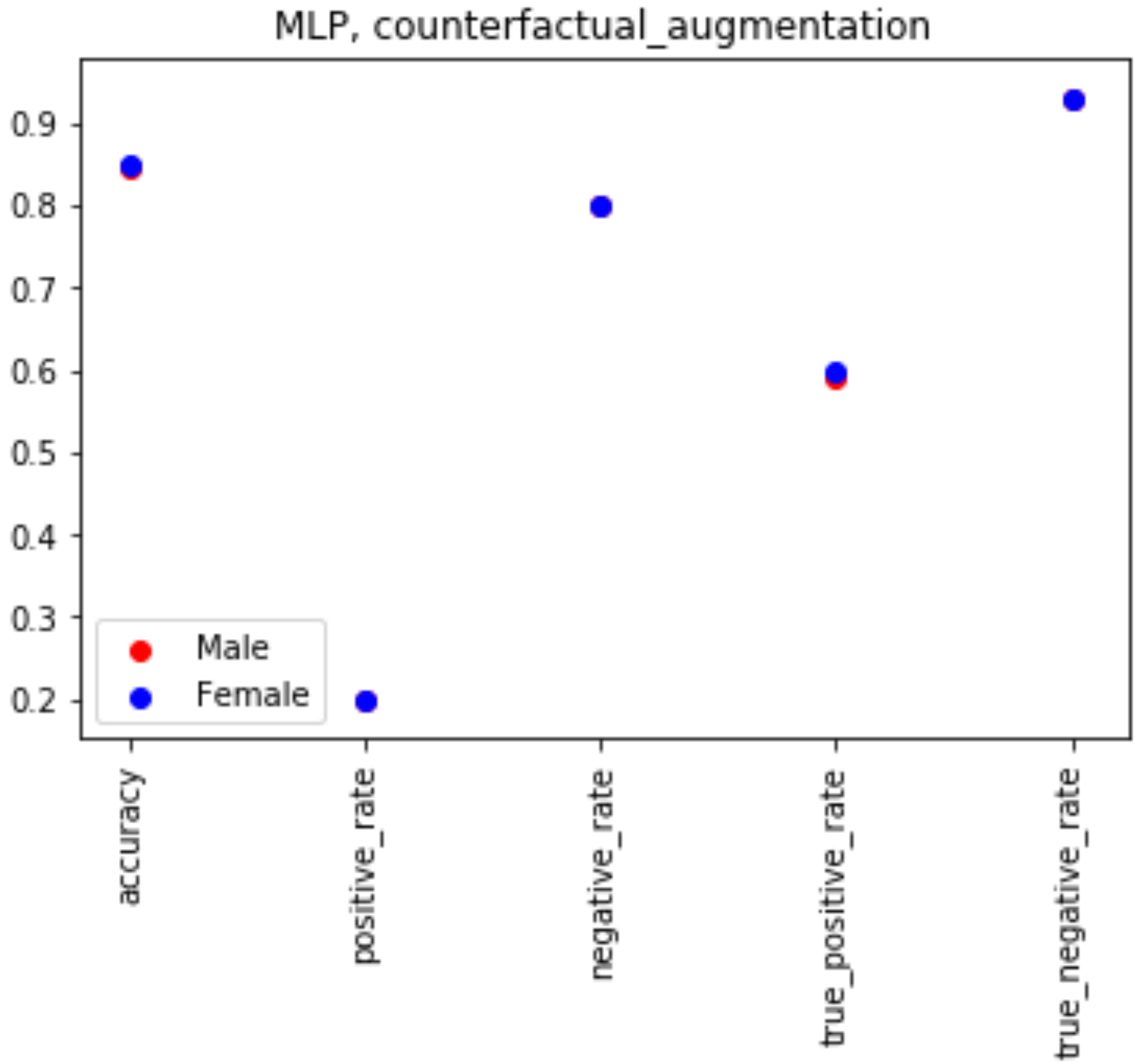
5.3 Augment data with counterfactuals

In [178]:
```
def with_gender_counterfacts(df):
    df_out = df.copy()
    df_out['sex'] = df_out['sex'].apply(lambda value: 1-value)
    result = pd.concat([df.copy(), df_out])
    return result
```

In [179]:
```
ctf_gender_augmented = with_gender_counterfacts(datav2)
(x_train, y_train), (x_test, y_test) = get_naive_dataset(ctf_gender_augmented)

predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
predictor.fit(x_train, y_train)
ctf_1 = evaluate_predictor_performance(predictor.predict(x_test), x_test, y_test)
model_summary("MLP, counterfactual_augmentation", "", ctf_1)
```

source:
Audace Nakeshimana & Maryam Najafian

5.3 Augment data with counterfactuals

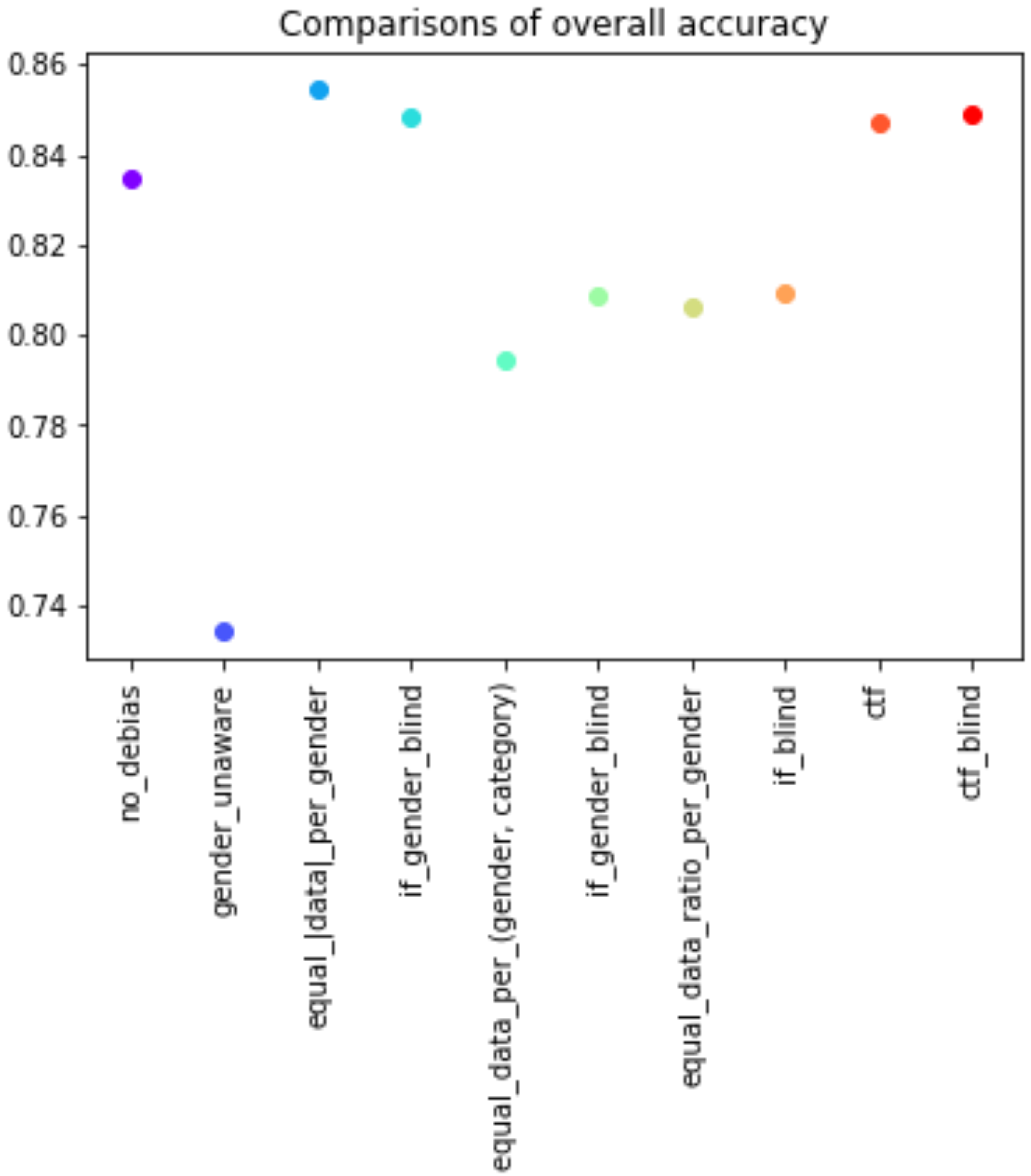


source:
Audace Nakeshimana & Maryam Najafian

5.4 Comparing Data-based approaches

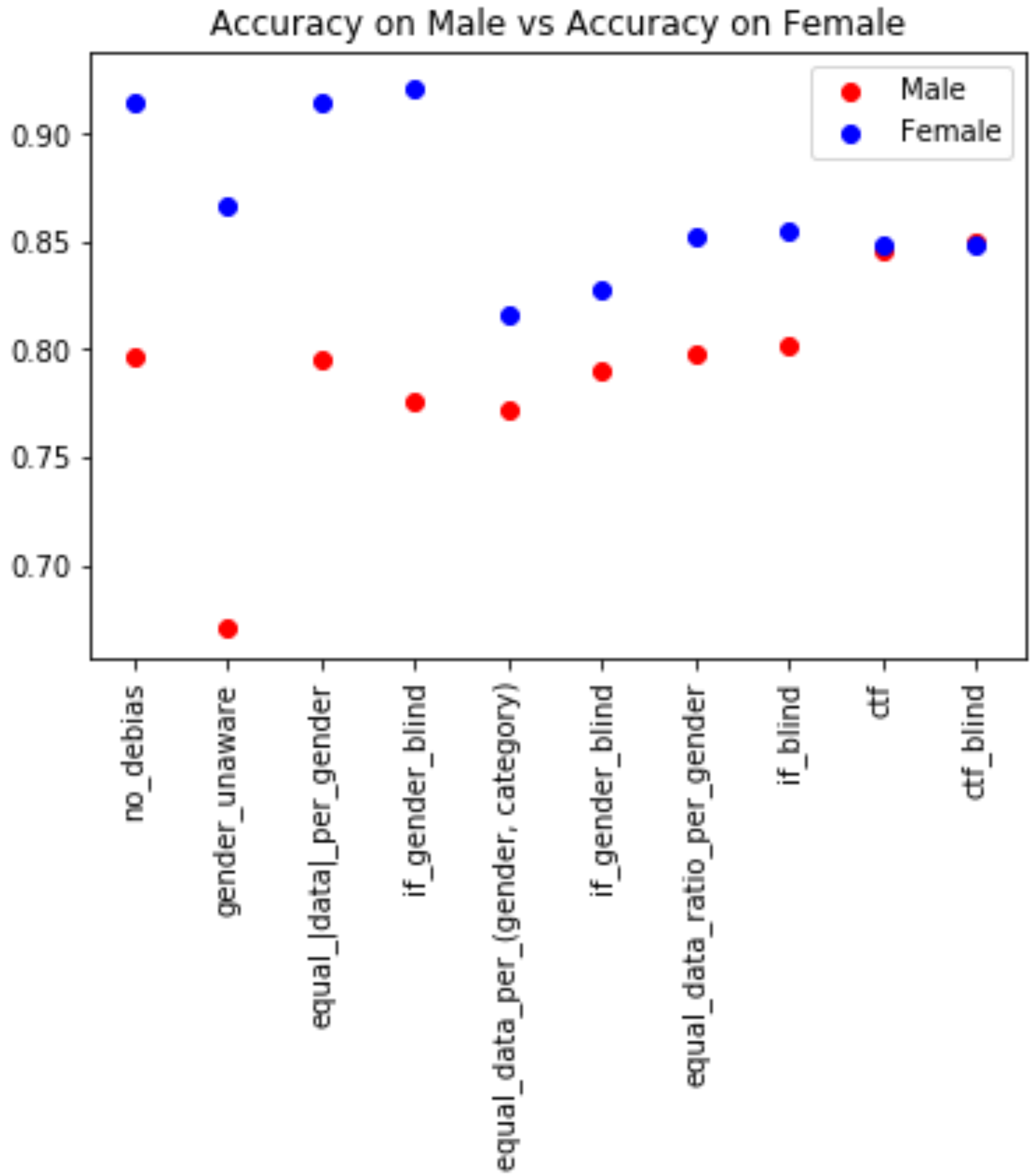
Let’s evaluate the metrics of interest on all approaches we’ve carried out so far.

Overall Accuracy



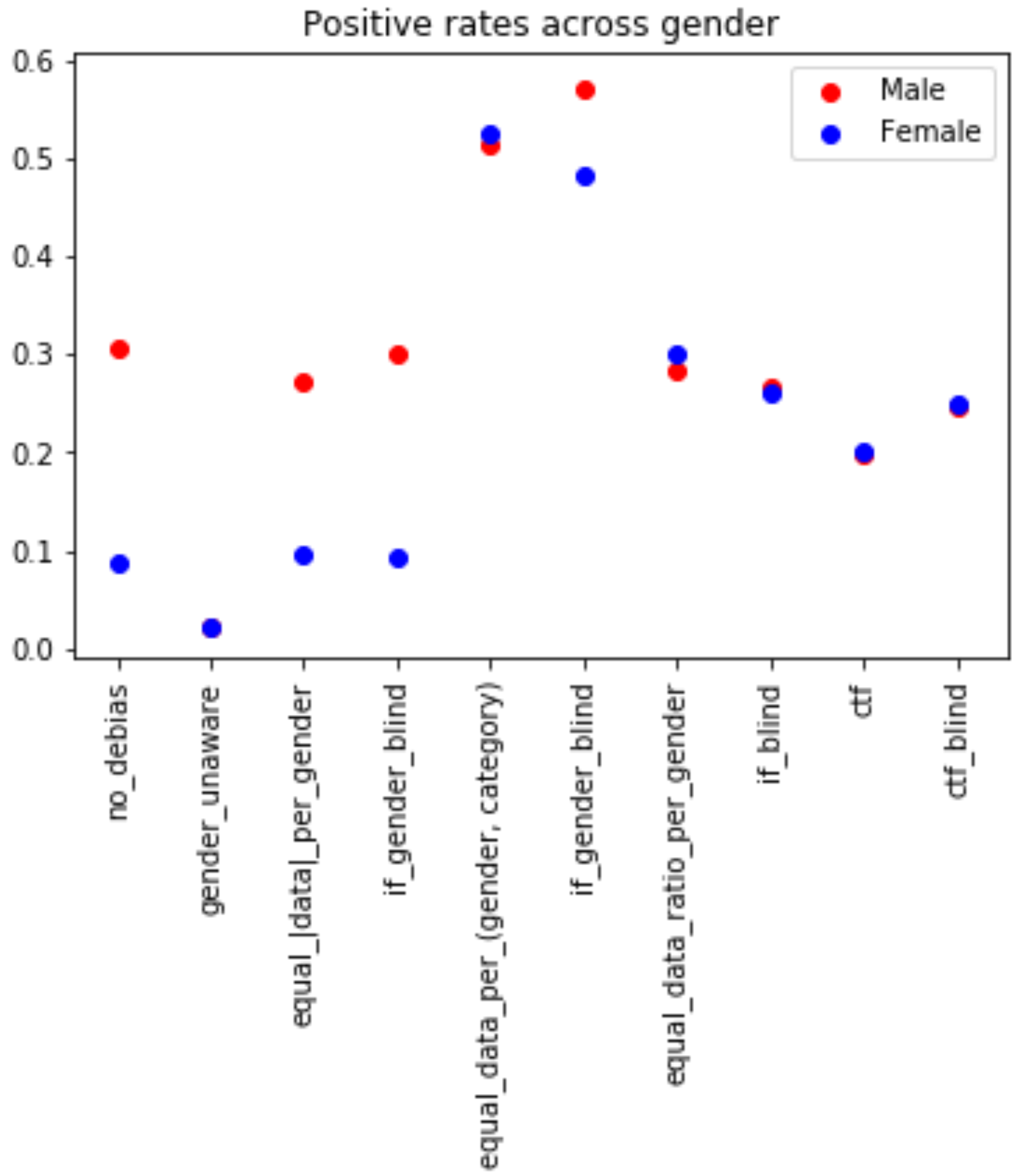
source:
Audace Nakeshimana & Maryam Najafian

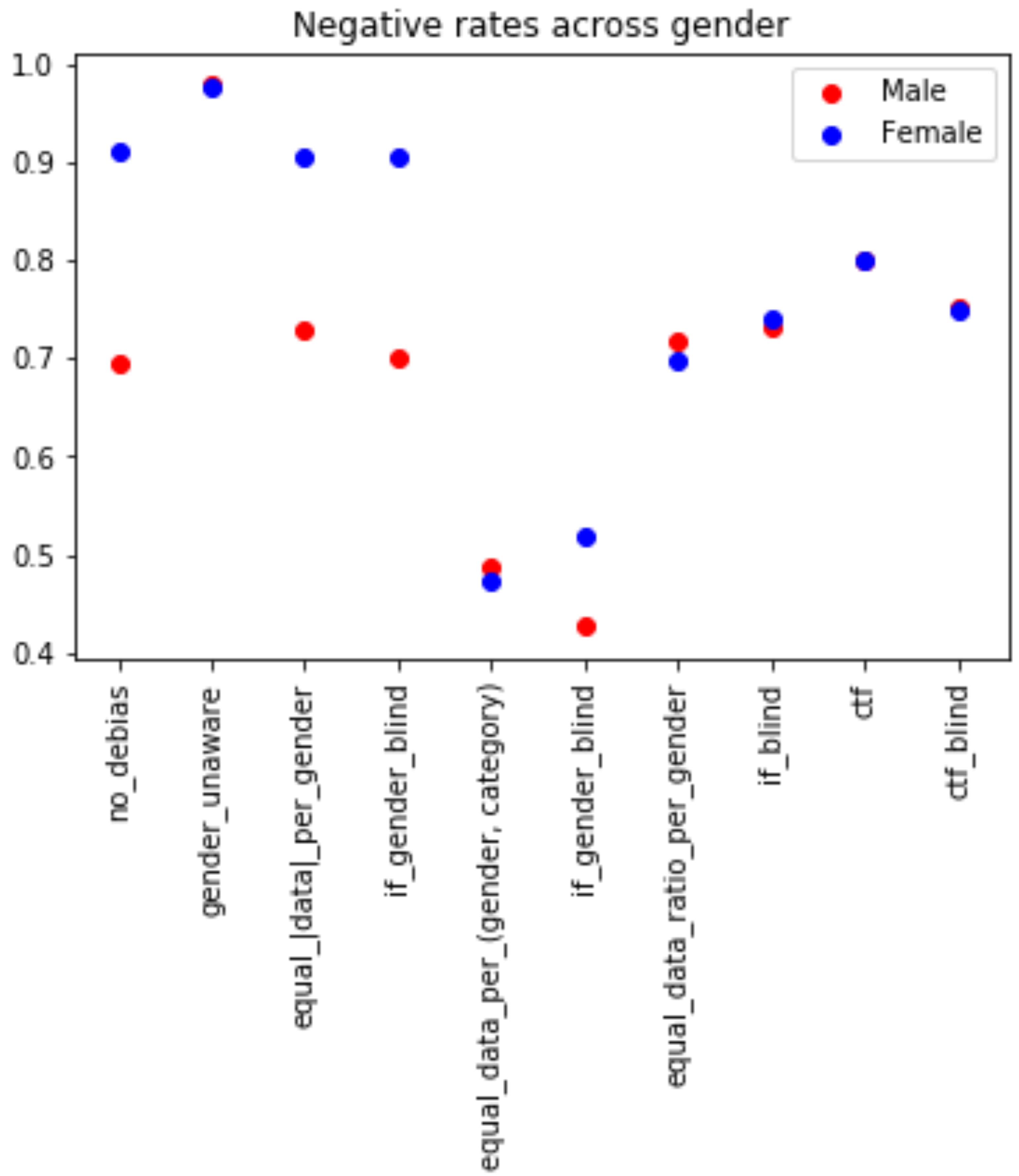
Accuracy across gender



source:
Audace Nakeshimana & Maryam Najafian

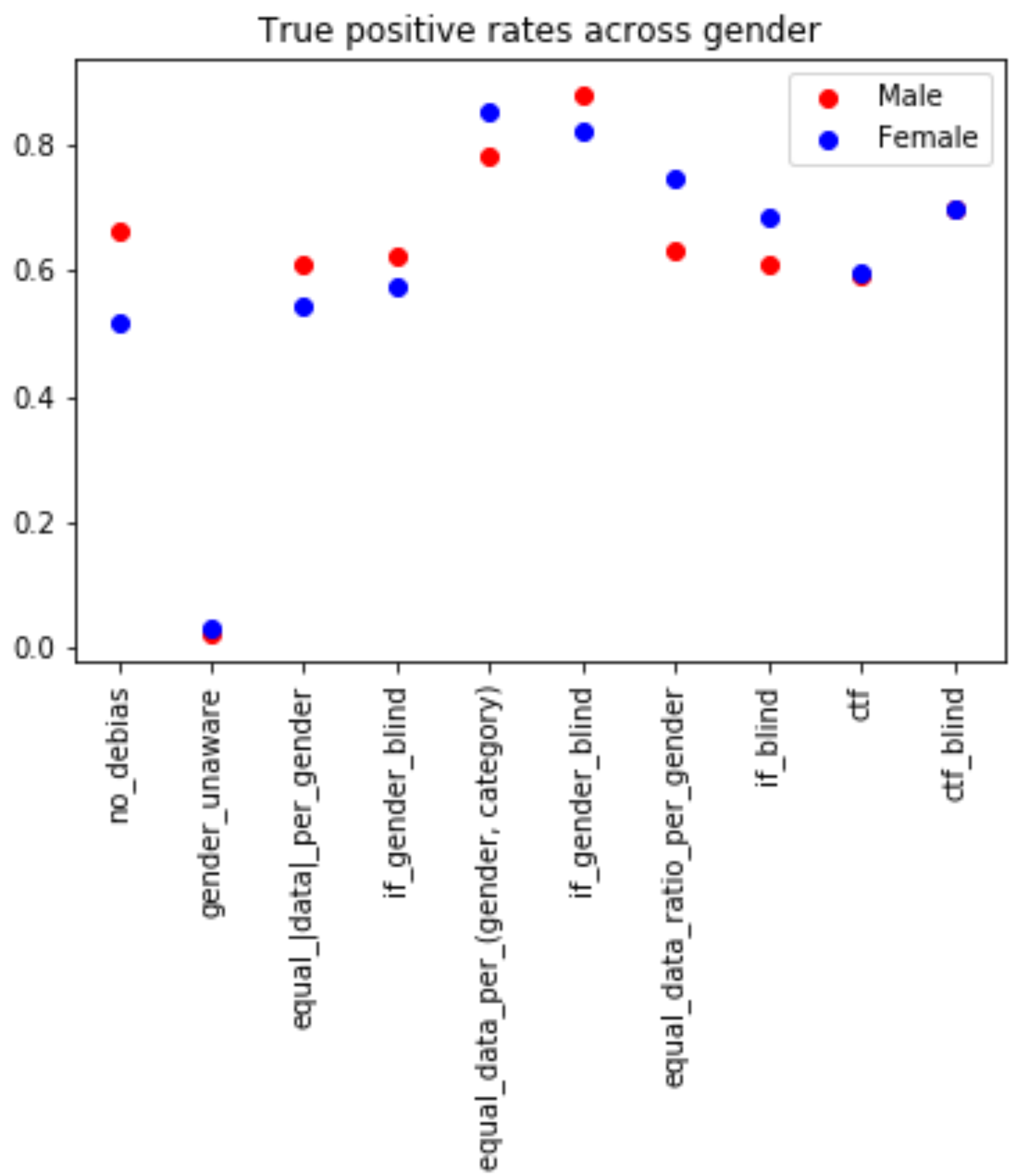
Positive and negative rates across gender

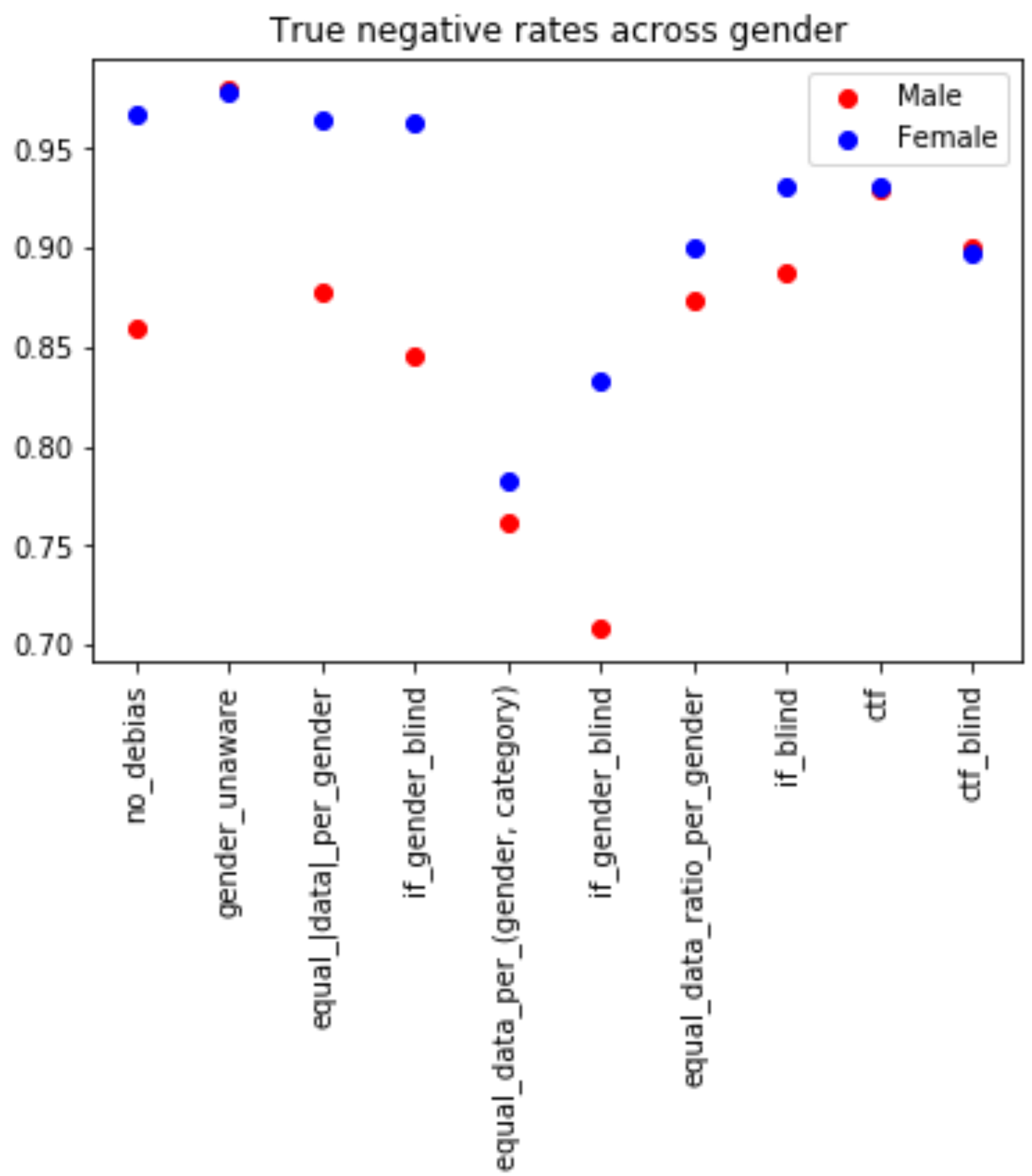




source:
Audace Nakeshimana & Maryam Najafian

True positive and true negative rates across gender





source:
Audace Nakeshimana & Maryam Najafian

Part 6:
Exploring Model-based Debiasing Techniques

We explore different model types and architectures to determine the least biased.

Motivation

- Different ML models inherently show different levels of bias.
- By changing the model type and architecture, we can observe which ones tend to be inherently less biased.

6.1 Single-model architectures

We use a model from each of the following model families:

Model Family	**Model We Used**
Support Vector Machines	sklearn.svm.SVC
Decision Tree Learning	sklearn.ensemble.RandomForestClassifier
Instance Based Learning	sklearn.neighbors.KNeighborsClassifier
Generalized Linear Models	sklearn.linear_model.LogisticRegression
Artificial Neural Network	sklearn.neural_network.MLPClassifier

source:
Audace Nakeshimana & Maryam Najafian

6.1 Single-model architectures

```
In [126]: lr = LogisticRegression(solver='lbfgs', multi_class='multinomial', random_state=1, max_iter=LR_MAX_ITER) # GLM
          rf = RandomForestClassifier(n_estimators=50, random_state=1) # Random Forest
          gnb = GaussianNB() # GLM
          mlp = MLPClassifier(max_iter=MLP_MAX_ITER)  # ANN
          svc = svm.SVC() # SVM
          knc = KNeighborsClassifier(n_neighbors=5)
          for model in [lr, rf, gnb, mlp, svc, knc]:
              model.fit(x_train, y_train)
```

Note: We use default parameters in most cases for simplicity and to stay within the scope of this module. In a more practical setting, we would use cross-validation and/or other hyperparameter search techniques to find the best parameters to use for each model.

source:
Audace Nakeshimana & Maryam Najafian

6.2 Multi-model architectures

Motivation: There is power in numbers. Let's train a group of different models on the same data, and then make a final prediction based on **consensus.**

We compared 2 consensus approaches:

- **Hard Voting:** Final prediction is the majority prediction among all models.
- **Soft Voting:** Final prediction is the average prediction.

6.2 Multi-model architectures

We leverage scikit-learn's VotingClassifier to combine single models

In [129]:
```
from sklearn.ensemble import VotingClassifier
```

In [130]:
```
def default_voting_classifier(voting='hard'):
    lr = LogisticRegression(solver='lbfgs', multi_class='multinomial', random_state=1, max_iter=LR_MAX_ITER)
    rf = RandomForestClassifier(n_estimators=50, random_state=1)
    gnb = GaussianNB()
    mlp = MLPClassifier(max_iter=MLP_MAX_ITER)
    svc = svm.SVC(probability = voting != 'hard')
    knc = KNeighborsClassifier(n_neighbors=5)
    voter = VotingClassifier(estimators=[('LR', lr), ('RF', rf), ('GNB', gnb), ('MLP', mlp), ('svc', svc)], voting=voti

    return voter
```

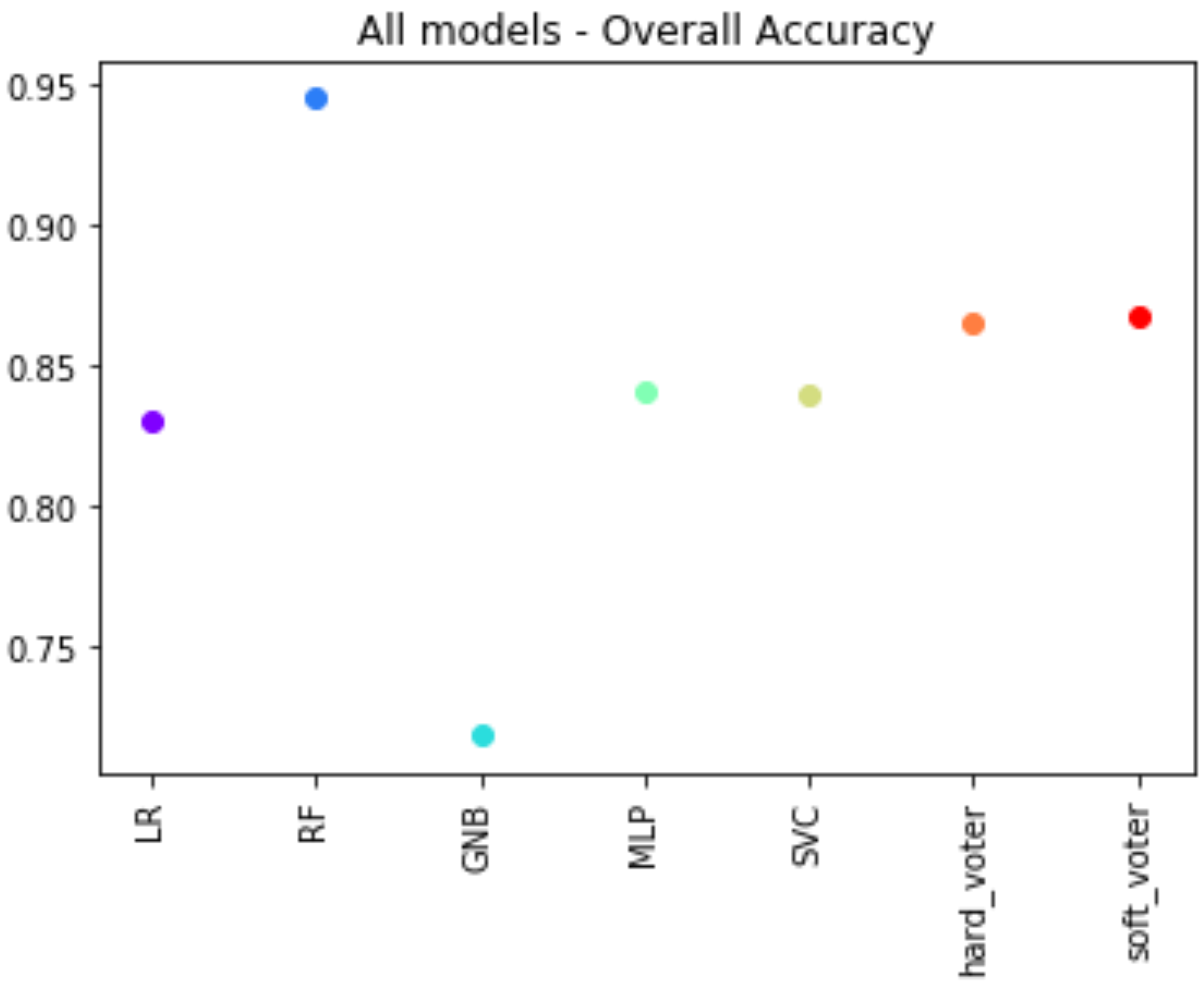
In [132]:
```
hardvoter = default_voting_classifier(voting='hard')
softvoter = default_voting_classifier(voting='soft')
for model in [hardvoter, softvoter]:
    model.fit(x_train, y_train)
```

source:
Audace Nakeshimana & Maryam Najafian

6.2 Comparing metrics across a single training session

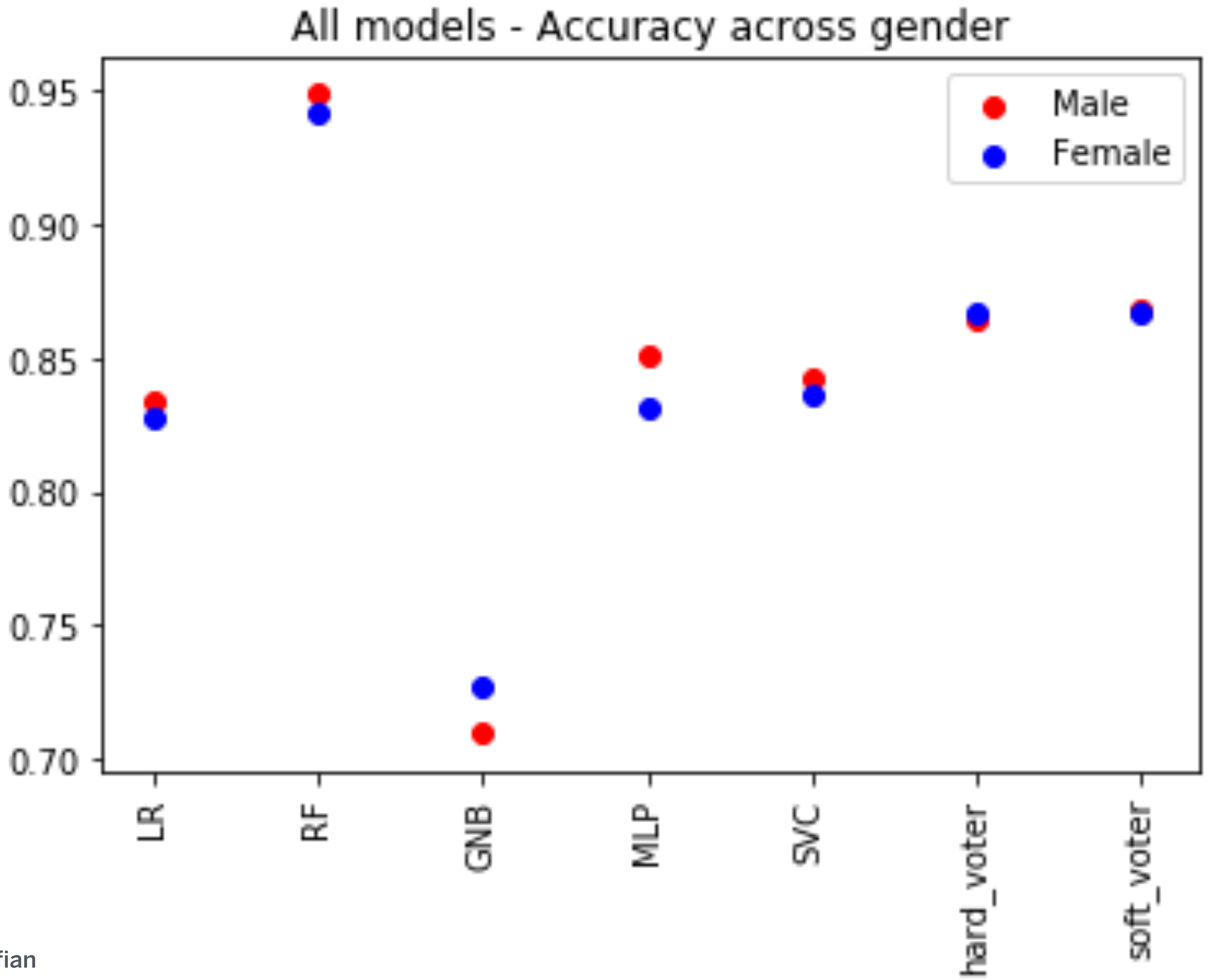
Let's evaluate the metrics of interest on all models that we've trained so far.

Overall accuracy



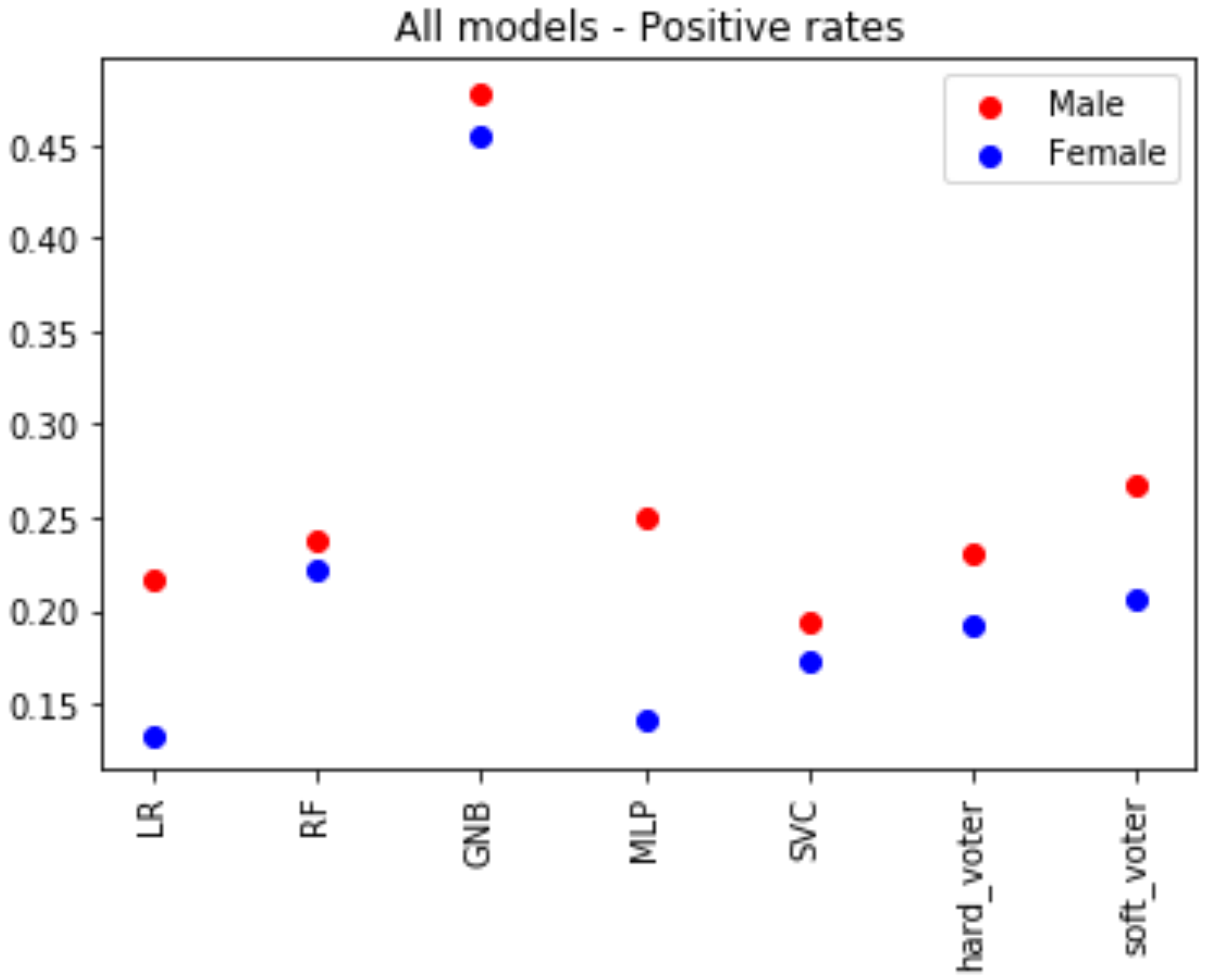
source:
Audace Nakeshimana & Maryam Najafian

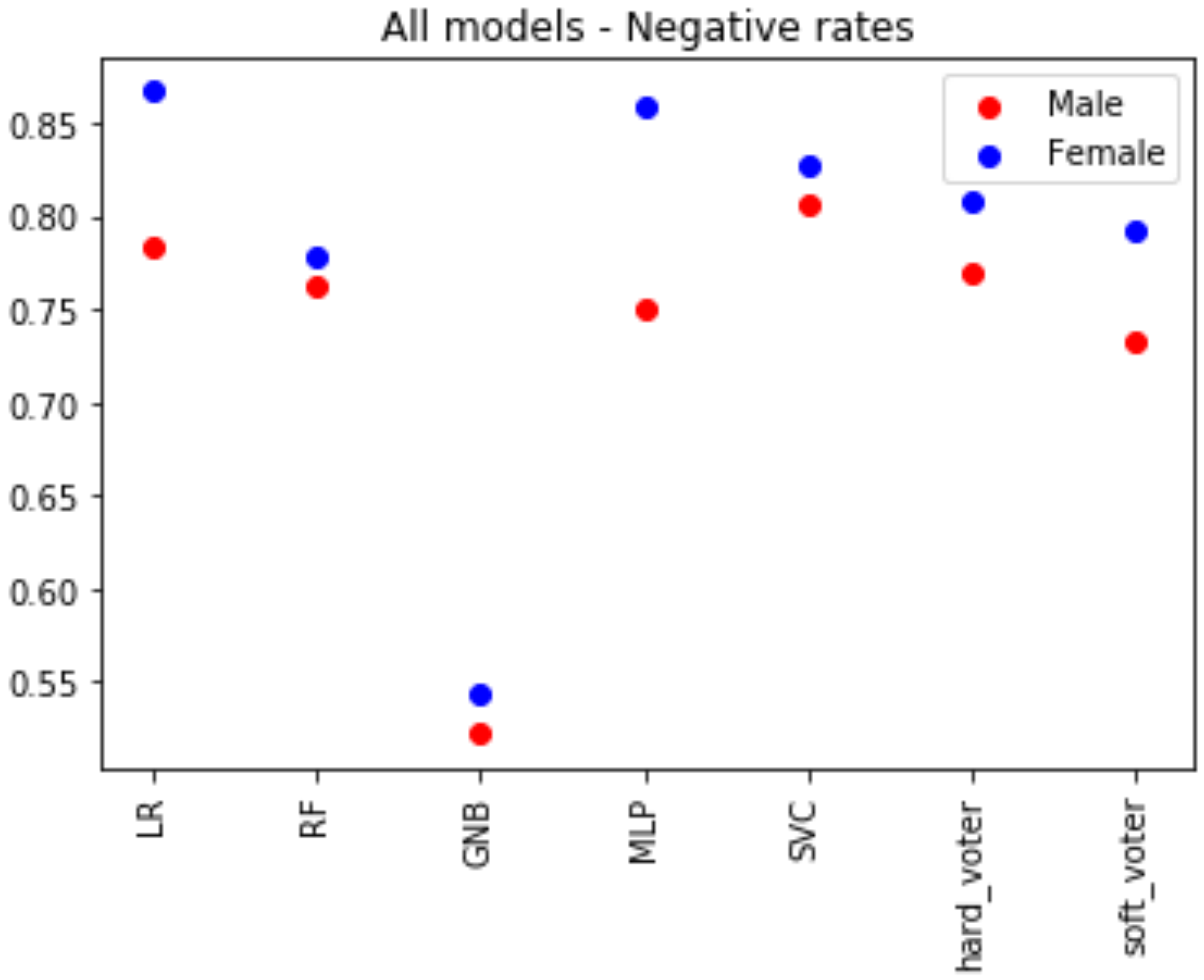
Accuracy across gender



source:
Audace Nakeshimana & Maryam Najafian

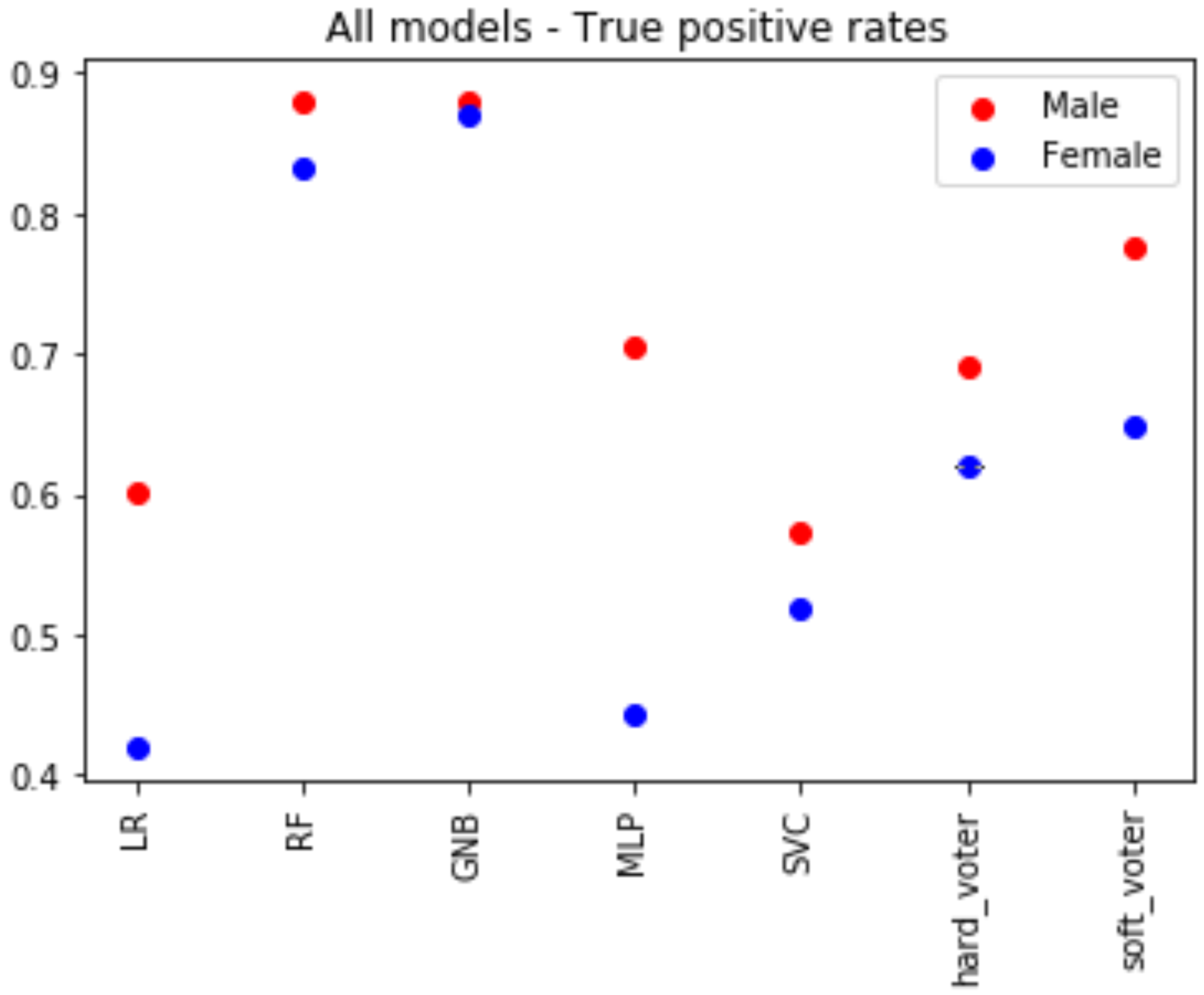
Positive and negative rates across gender





source:
Audace Nakeshimana & Maryam Najafian

True positive and true negative rates across gender



source:
Audace Nakeshimana & Maryam Najafian

6.3 Comparing metrics across multiple training sessions

Motivation:

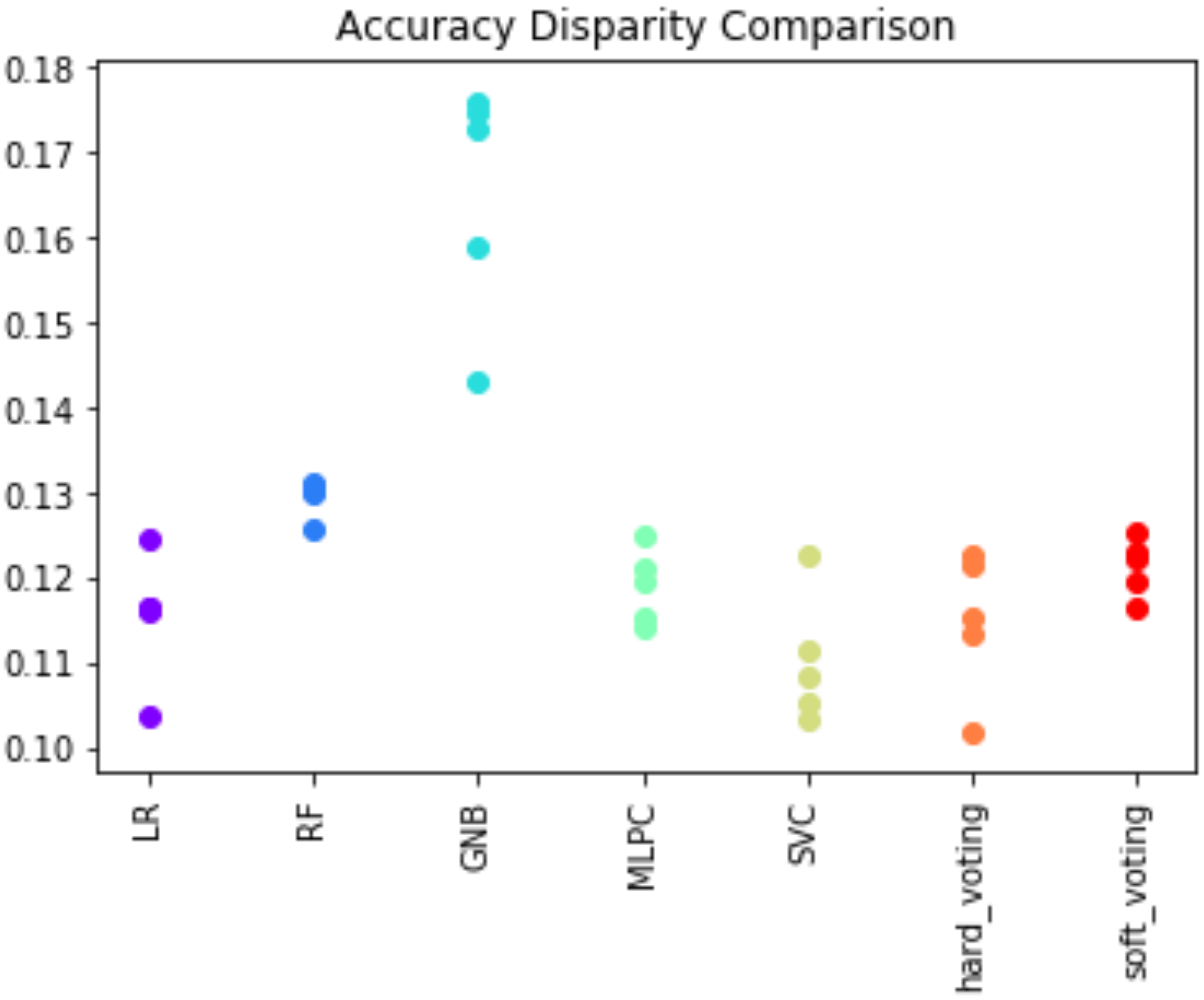
Due to randomness, a single training session does not say much about the general model behavior. We should therefore run multiple sessions to get a better understanding of average behavior.

6.3 Comparing metrics across multiple training sessions

Approach:

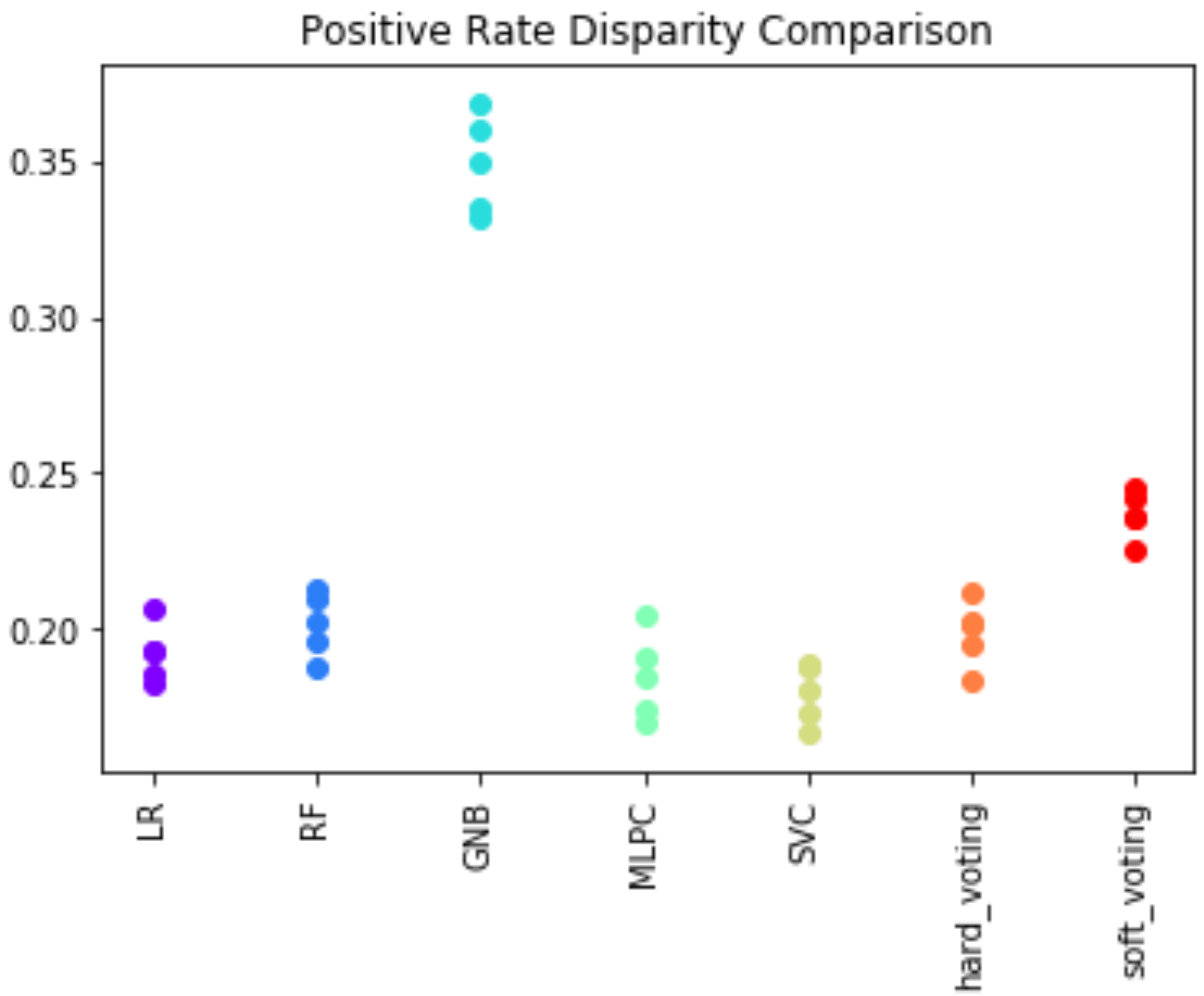
For each model type, train 5 instances of this type on the data. Then, for each instance, evaluate the absolute value of the difference in metric of interest between male and female demographics from the test data.

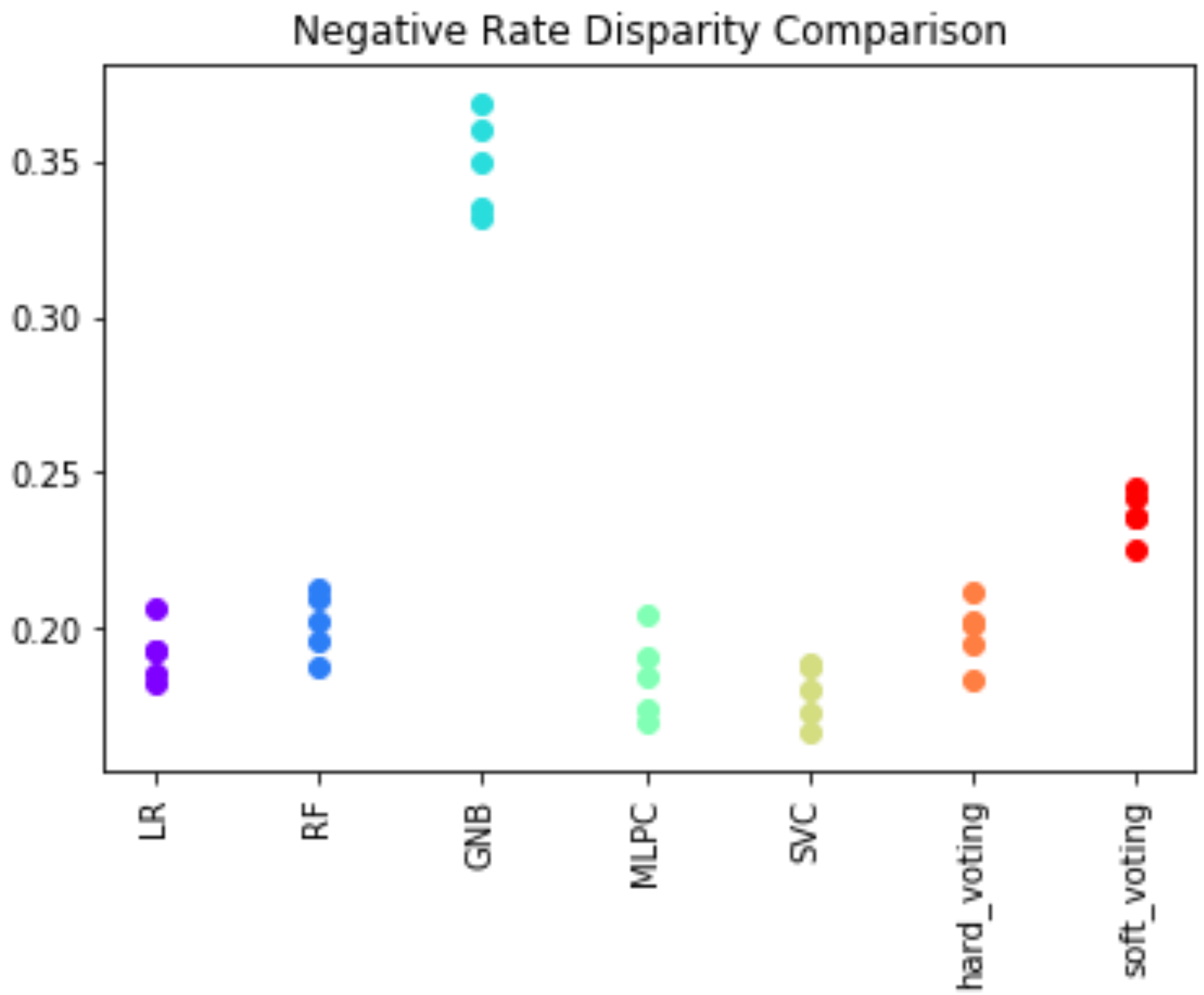
Accuracy disparity comparison



source:
Audace Nakeshimana & Maryam Najafian

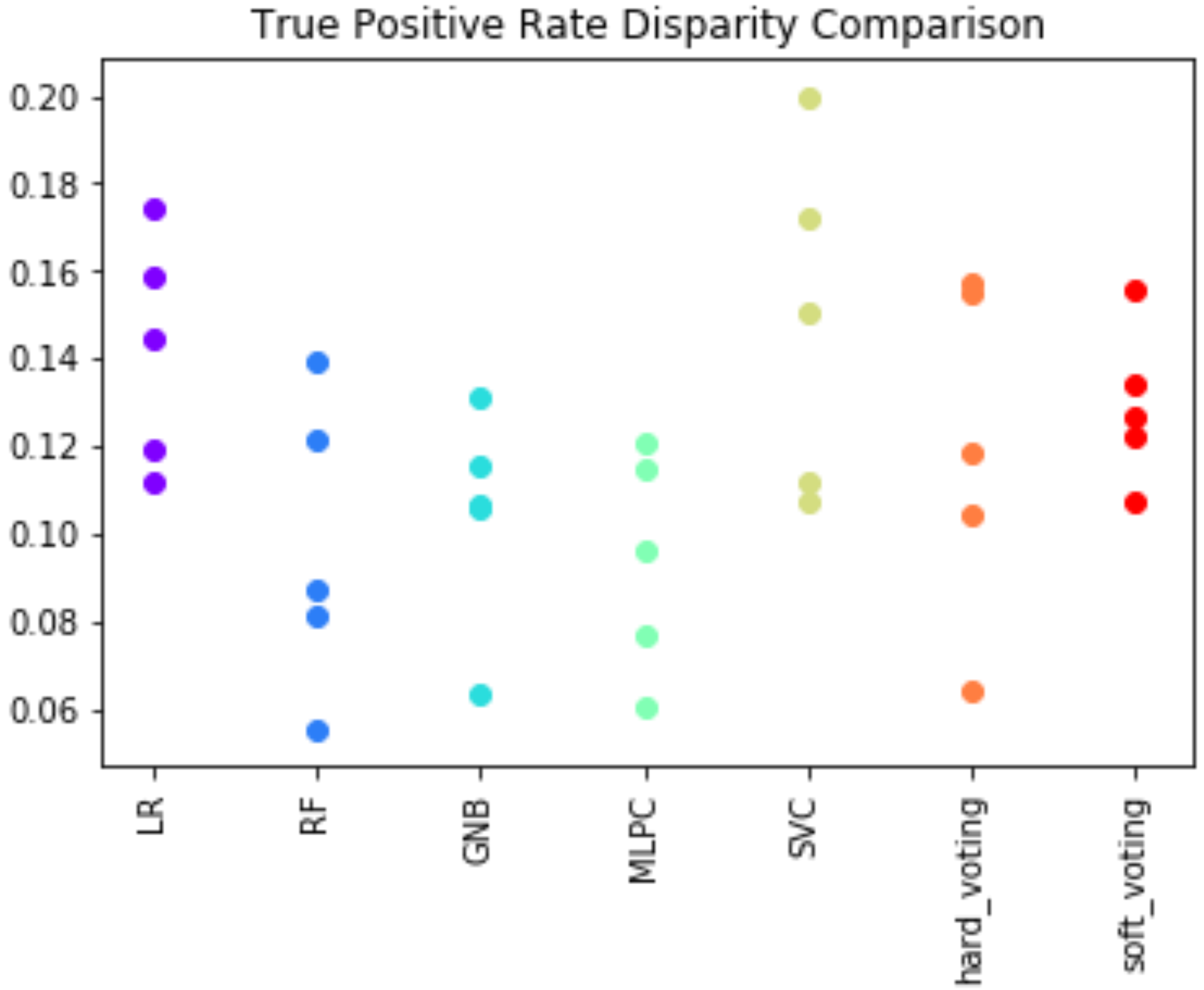
Positive and negative rate disparity comparison

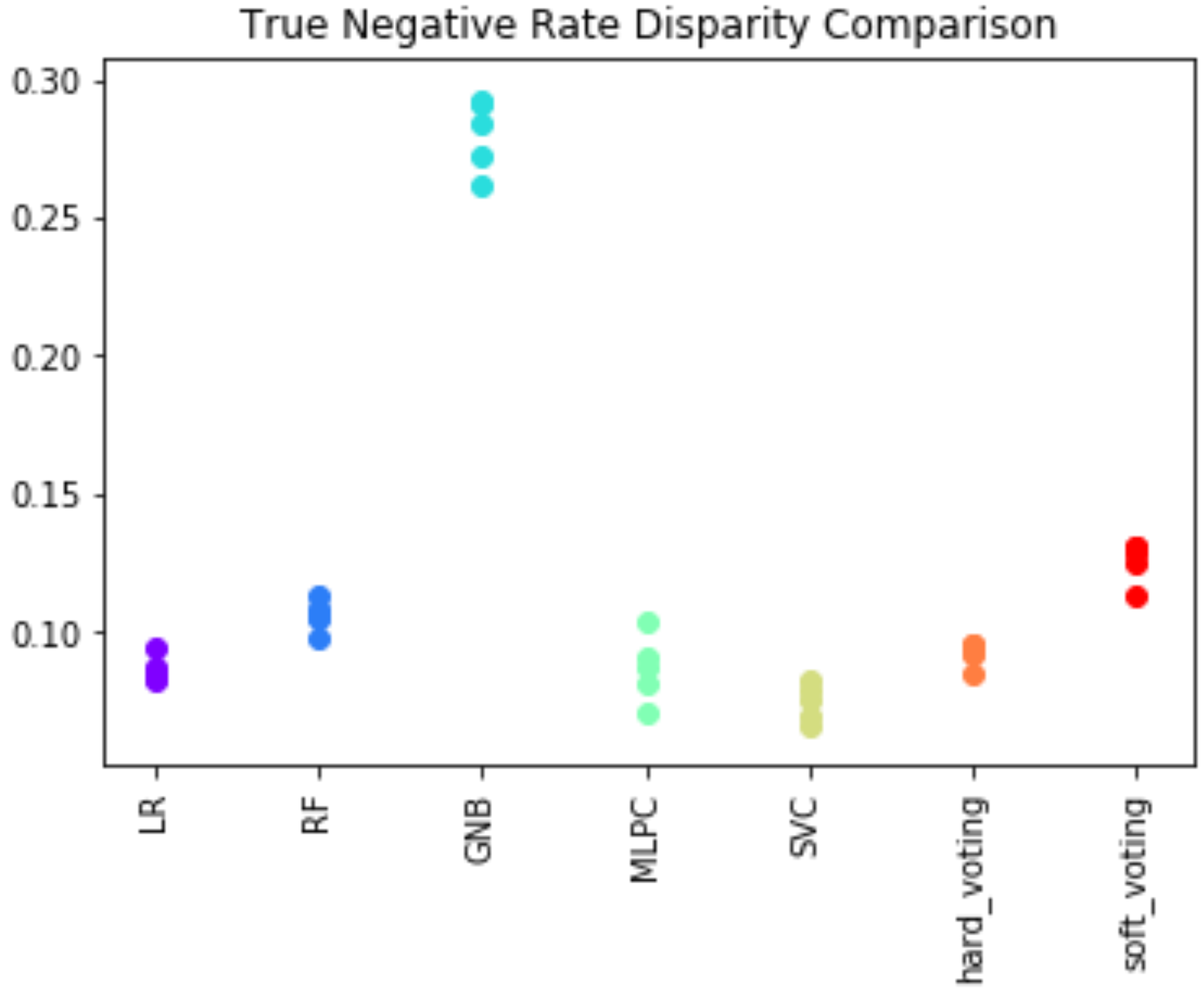




source:
Audace Nakeshimana & Maryam Najafian

True positive and true negative rate disparity comparison





source:
Audace Nakeshimana & Maryam Najafian

Part 7: Conclusion

Next steps for strengthening our understanding & application of ethics in ML.

Suggested next steps

- Checkout repository for the module at:

https://github.com/heyaudace/ml-bias-fairness

- Explore more advanced debiasing techniques.
- Share & discuss across your team, organization, community, etc.

Thank you

Audace Nakeshimana
Undergraduate Student and Researcher, MIT

audace@mit.edu

Maryam Najafian
Advisor & Research Scientist, MIT

najafian@csail.mit.edu

References

- Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [http://archive.ics.uci.edu/ml]. Irvine, CA: University of California, School of Information and Computer Science.

- Bishop, Christopher M. Pattern Recognition in Machine Learning. New York: Springer 2006

- Hardt, Moritz.(2016, October 7). Equality of Opportunity in Machine Learning. Retrieved from https://ai.googleblog.com/2016/10/equality-of-opportunity-in-machine.html

- Zhong, Ziyuan. (2018, October 21). A tutorial on fairness in Machine Learning. Retrieved from https://towardsdatascience.com/a-tutorial-on-fairness-in-machine-learning-3ff8ba1040cb

- Kun, Jeremy.(2015, October 19). One definition of algorithmic fairness: statistical parity. Retrieved from https://jeremykun.com/2015/10/19/one-definition-of-algorithmic-fairness-statistical-parity/

- Olteanu, Alex.(2018, January 3). Tutorial: Learning Curves for Machine Learning in Python. Retrieved from https://www.dataquest.io/blog/learning-curves-machine-learning/

- [Garg et al. 2018] Garg, S.; Perot, V.; Limtiaco, N.; Taly, A.; Chi, E. H.; and Beutel, A. 2018. Counterfactual fairness in text classification through robustness. arXiv preprint arXiv:1809.10610.

- Wikipedia contributors. (2019, September 6). Algorithmic bias. In Wikipedia, The Free Encyclopedia. Retrieved 07:23, September 12, 2019, from https://en.wikipedia.org/w/index.php?title=Algorithmic_bias&oldid=914352968