University of Hertfordshire UH

UNIVERSITY OF HERTFORDSHIRE

Department of Computer Science

MSc Data Science and Analytics with Advanced Research

7COM1075 Data Science and Analytics Masters Project

3rd September 2021

Project Title

Auditing Bias and Fairness in Data Science Systems

Name: Mohammed Saaduddin Ahmed

ID Number: 19001506

Supervisor: Prof. Deepak Pandey

MSc Final Project Declaration

This report is submitted in partial fulfilment of the requirement for the degree of Master of Science in Data Science and Analytics with Advanced Research at the University of Hertfordshire (UH).

It is my own work except where indicated in the report.

I did not use human participants in my MSc Project.

I hereby give permission for the report to be made available on the university website provided the source is acknowledged.

Signed: Mohammed Saaduddin Ahmed

ID Number: 19001506

Abstract

Due to the rapid emergence of machine learning technology, many people are interested in its impact on society. This paper aims to analyze the various aspects of supervised learning and how it can be utilized to improve the quality of life. Learning algorithms can often predict future events with high accuracy. An algorithm could be used to predict who will pay back their loan. It could also help determine who should get a loan in the first place. Unfortunately, even the best predictors can make mistakes. To prevent these types of errors, it is essential to train the system correctly. For a group that has little data on its own, it is more likely to make poor predictions than the general population. A mortgage loan predictor might end up profiling people in a group as being at high risk of default even though they have paid their loan. This could result in them accruing a negative credit score even though they have paid their loan. Despite the need for this type of machine learning, no widely available framework can prevent discrimination based on sensitive characteristics. This paper aims to provide a framework that can detect this type of discrimination. The idea of fairness through unawareness is misguided due to the existence of redundant encodings. Another approach is demographic parity, where the prediction is not related to the sensitive attribute. For instance, when predicting an event that is correlated with a group's membership, this approach is not desirable. Although women are more likely to develop heart failure than men, the incidence of heart failure among women is still higher than among men. We introduce a framework that helps identify potential discrimination concerns based on sensitive attributes. We believe that individuals should have the same chance of being classified as achieving a desirable outcome. This principle is known as equality of opportunity. Our framework aims to improve the incentives for decision-makers to make better predictions by shifting the cost of poor decisions from the individual to the maker. The goal of improving the accuracy of prediction is well aligned with the goal to avoid discrimination. This project explores fairness in machine learning by looking at techniques for mitigating bias. We start by illustrating bias in machine learning, and then we will look at techniques for mitigating bias

Table of Contents

List of Tables

List of Figures

1.0 Introduction

To be fair and unbiased is a prized human value. A society in which people can thrive must be united through practices and institutions that are considered fair. History has been arguing about what fairness means, and there have been debates to address regional and gender inequalities in the UK. Several complex fairness issues triggered by the COVID19 pandemic have made fairness and equality the centre of public debate (Truss, 2020). Although inequality and inequity have complex causes, bias in the decision-making organizations about individuals is often an important aspect. The impacts of efforts to combat undue bias in decision making are often not measured or are painfully delayed becoming effective, but the decision making is now facing a time of change. Data usage and automation have existed in some areas for many years but are now rapidly expanding due to the dramatic increase in the amount of data available, and machine learning algorithms have become more sophisticated and accessible. Data provides a powerful weapon for determining where biases occur and measuring whether those mitigation efforts are practical. If an organization has solid data on how people are treated, it can construct insights into what is causing those differences and try to address them. This report addresses the bias present in the machine learning models, which are being used by the banking, insurance, and pension industry to predict the income levels of an individual.

For example, given the data, many lending institutions are trying to rely on these models to predict if an individual will default on a loan (Liang, 2011) Now because machine learning is being used in high-stakes applications, errors that purpose it to be unfair can cause discrimination, preventing positive demographic groups from getting access to fair loans. According to Prater (2010), credit score card agencies withinside the US must observe the Credit Card Act of 2009 and sooner or later, each time an applicant applies for a brand-new credit card or for increasing the restriction on a present credit card, there could be thorough check-up of income and property of the applicant to check whether the applicant can payback. According to this law, Federal Reserve allowed issuers of spot credit score facility providers (including retail stores) to apply statistical profits estimation models trained with the aid of using the credit score reporting bureaus. The credit model is executed to decrease default by the borrowers and restrict their capability to strain the banking/monetary system (Prater, 2010). Since machine learning models are increasingly being used in making important decisions that affect human lives, it is vital to ensure that the prediction is not biased towards any protected attribute such as race, sex, age, marital status, etc. (Biswas and Rajan, 2020).

1.1 Research Questions

Following are the research questions that we will be exploring in this report

- Do machine learning algorithms inherently exhibit bias?
- How do we measure if bias/discrimination exist in our model, and if they exist how can we rectify those errors?
- How can we be sure the Machine learning decisions are free from bias and or discrimination?
- Does bias mitigation improve the income classification of the adult women in United States census of 1994?

1.2 Aims and Objectives

Following are the aims and objectives of the project

- To explore and understand the different steps and various principles employed in building a fair machine learning model.
- Explore data-based techniques to mitigate bias in machine learning
- Explore model-based techniques to mitigate bias in machine learning
- Apply the techniques for mitigating bias as mentioned above on the UCI Adult data set to remove gender bias in the income category predictions.

1.3 Overview

The literature review section of the report provides a foundation for the study, which is then used to discuss the results and interpretations. The literature review is the most critical component of this dissertation. The Methodology section in the report helps readers understand how the study will be carried out. This section also provides a clear understanding of the procedures that will be used during the study. The experiment and results section of the report describes the various techniques and tools used to analyze and summarize the data. It also includes an introduction to statistical devices and methods. The evaluation section of the report summarizes the study's findings and provides a report on the results. It does so by organizing the data in a variety of ways. The study's findings are discussed in the conclusion section of the project and their implications for various aspects of practice and future research. All the references and the code are included in the reference section and the report's appendix section, respectively.

1.4 Management of the project

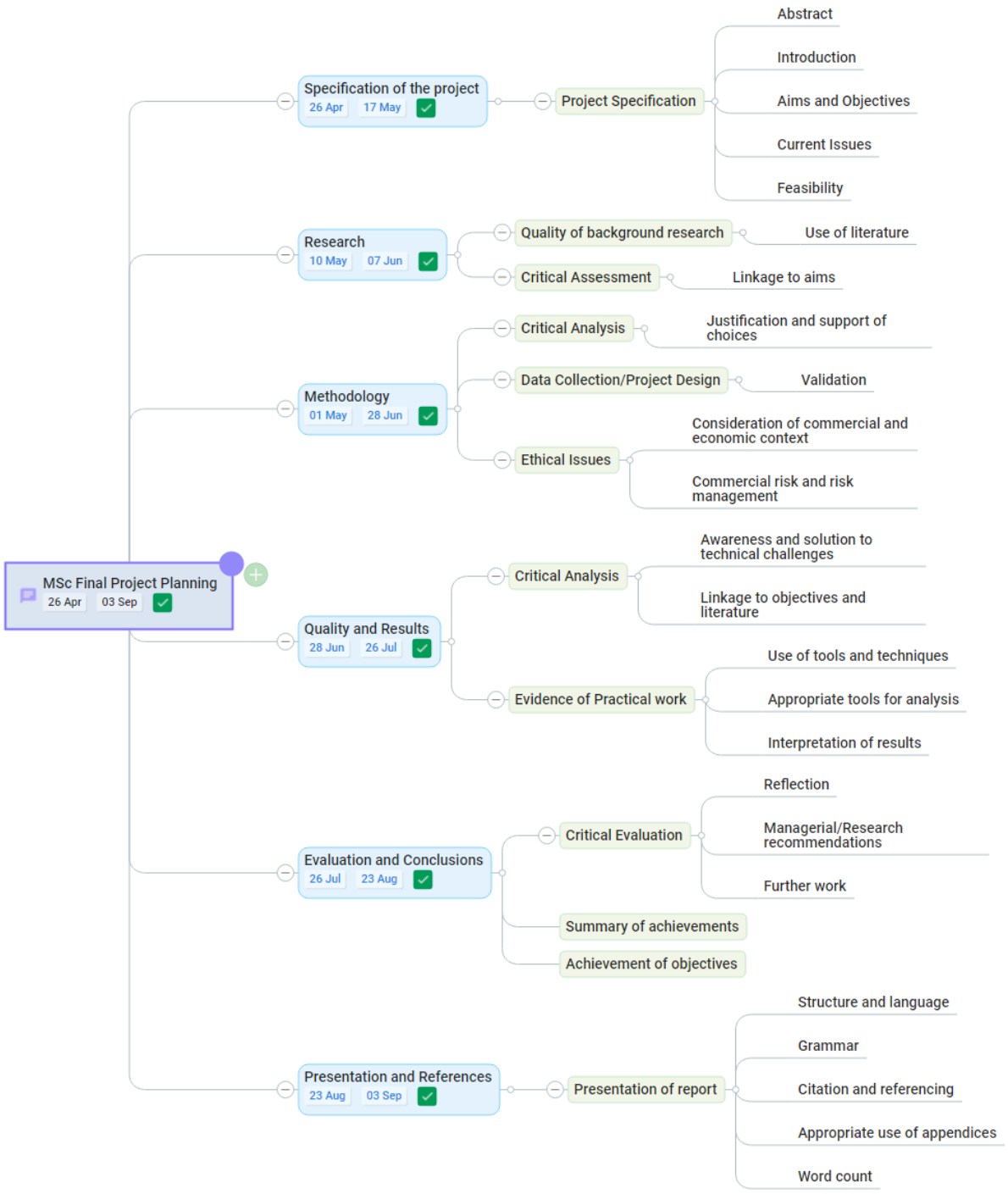

Figure 1: Project Plan

2.0 Literature Review

Due to the increasing use of algorithms for decision making, it is imperative that they are not just algorithms but also treat all users fairly (Baeza-Yates, 2018). Bias can be defined as unfair or systematic discrimination against a group or individuals. Technical bias and emergent bias are two types of biases that can affect a computing system. The former refers to the biases that can affect the system's design, while the latter focuses on the factors that can affect its operation. Emergent bias and technical bias are two types of biases that can affect a computing system. The former refers to the biases that can affect the system's design, while the latter focuses on the factors that can affect its operation (Friedman & Nissenbaum, 1996). From an algorithmic standpoint, bias can arise from different sources: data or bias in an algorithm. Even when an algorithm is not biased, its output might still be biased. A feedback loop in machine learning systems is a mechanism that enables learning models to generate biased outputs (Baeza-Yates, 2018). An algorithmic bias can result in unfair predictions for users of different groups. This form of bias can be considered a form of discrimination. Fair and equal treatment is the concept that everyone is treated equally. This concept is why the term biased is often used interchangeably with unfair and unbiased with fair (Verma & Rubin, 2018).

Machine learning models are used in various domains, such as risk modelling, decision making, and education. They can also help identify biases in the decisions they make. Enhancing transparency makes the model more interpretable and increases the confidence in using machine-learning systems for decision making (Zhou & Chen, 2018). In machine learning, fairness is a crucial criterion that has gained significant attention in recent years. Various ways were presented in the paper to achieve fairness in machine learning. By modifying the input data, adapting algorithms and adjusting the outputs/predictions of a machine learning model, we can improve its performance (Abdollahi and Nasraoui, 2018).

Fairness in machine learning is a concept that refers to treating users from various groups equally in terms of the input data. Simply ignoring the sensitive features of machine learning models is not enough to eliminate bias. Doing so could make the models less biased, but it would still not eliminate the biases (Pedreshi, Ruggieri and Turini, 2008). Various approaches have been suggested to reduce or obfuscate sensitive attributes while retaining good overall performance/accuracy (Zemel et al., 2013.; Kamiran et al., 2012). Several researchers provided a fair algorithm overview and definitions of fairness in machine learning (Burke, Sonboli and Orgonez-Gauger, 2018; Doshi-Velez and Kim, 2017). (Kamishima et al., 2012) presents a regularization approach that aims to remove bias from the classification model. It can be used for the development of other probabilistic models. (Fish et al., 2016) proposed a method

that shifts the decision boundary between accuracy and bias in the machine learning algorithm.

A biased data input perspective can be used to evaluate the extent to which discrimination occurs in the context of classifications. In particular, (Pedreshi et al., 2008) introduced the concept of discriminatory classification rules. (Kamiran and Calders, 2012) proposed a method to achieve fairness by converting partial data into unbiased data by dropping the sensitive attributes. (Lum and Johndrow, 2016) provide a probabilistic framework for the removal of bias from predictive models based on the specific data types. An excellent first step in addressing the potential unfairness of an algorithmic decision is defining the potential unfairness of the solution. This step can be followed by quantifying the potential unfairness of the algorithm.

Data-based models are often built based on the quality of the input data. This is because they rely on the patterns and insights that the data provides to develop suitable solutions. The first step in this process is determining the target set of adjustments to make the model work seamlessly. For this pre-processing, it can be used to reweigh the input data (Kamiran and Calders, 2012), or it can be used to reduce discrimination by learning fair representations of the original data (Feldman et al., 2015), or encoding protected attributes (Zemel et al., 2013). The second type of bias mitigation strategy focuses on making algorithmic adjustments. This method works by looking at the algorithm's fairness as an optimization problem and then solving it in-processing (Kamishima et al., 2012). Another type of bias mitigation strategy involves considering the already calculated solutions. This method can be used for any classification outcome and alter the classification threshold for specific groups or instances (Hardt, Price and Srebro, 2016). Our approach can provide the most accurate representation of fairness in terms of correctness. Before a machine learning model is deployed, it should be evaluated for its accuracy and precision. Also, it should be noted that it should not have bias. The performance of a machine learning model in real-world conditions is different from that of its trained counterpart. It should be analyzed and monitored regularly to prevent bias (Robinson, 2020).

2.1 Legal, Social, Ethical and Professional issues

AI is becoming more prevalent in the private and public sectors as they increasingly turn to machine learning algorithms and systems to improve decision-making. The rapid emergence and evolution of these technologies have disrupted various economic sectors. Before machine learning, people and organizations made various decisions such as hiring, advertising, and criminal sentencing. These decisions were often based on local and federal laws. Today, many of these decisions are being

made by machines equipped with statistical rigour and scale. Due to the nature of machines' artificial intelligence, it is starting to become clear that their decisions may not be as objective as previously thought. For instance, some algorithms may be prone to generating biases, such as those used by courts in the U.S. With the emergence of algorithmic bias in various applications, operators and other concerned parties must take responsibility for addressing this issue. This paper proposes various mitigation steps designed to address this issue and prevent harmful impacts on users.

The COMPAS algorithm, used by courts to determine whether a defendant should be detained or released pending trial, was allegedly biased against African Americans. The algorithm uses a risk score to assign a defendant's likelihood of committing an offence in the future. African Americans were more likely than whites to get a higher-risk score, which resulted in more extended periods of detention (Corbett-Davies et al., 2017).

Joy Buolamwini, a researcher at MIT, discovered that the algorithms used to develop facial recognition software failed to recognize darker-coloured faces. In her study, the software was able to identify a person with a white face 99 per cent of the time (Hardesty, 2018).

Google's results were more likely to return ads for people with arrest records than white names when searching for African American names. Similarly, the same query results were returned for people with higher-interest credit cards (Sweeney, 2013).

Amazon, which has a male workforce of 60 per cent and a female workforce of 74 per cent, stopped using an AI-powered recruiting algorithm after finding gender bias in its data. The company sourced its data from over a decade of applications. The algorithm was developed by engineers who were mainly white males (Vincent, 2018).

These examples show how biases can happen even without malicious intent. They also show how these results can be achieved through various means without validation or approval by the algorithm's creators.

2.2 Commercial and economic issues

In 2018, President Donald Trump signed legislation that encourages algorithms in the criminal justice system. The goal of these systems is to determine which inmates can get reduced sentences for completing their rehabilitation or educational programs (Lee, Resnick and Barton, 2019). The report emphasizes the importance of having a process that ensures that the algorithms are designed and executed relatively and ethically. Some decisions may be best served by algorithms and AI tools, while others require more thought and analysis. Further, testing and reviewing specific algorithms can help identify and mitigate discriminatory outcomes.

3.0 Methodology

This section of the report contains the methodology employed during the project. This project explores fairness in machine learning by looking at the techniques for mitigating bias. We start by illustrating bias in machine learning, and then we look at techniques for mitigating bias. Specifically, we will explore two types of techniques. Data-based techniques to calibrate and augment our data set to mitigate bias in machine learning and Model-based techniques will explore different models and architectures that will assist us in achieving a less biased model, as shown in Figure 1. We carried out these techniques to the UCI Adult data set to mitigate gender bias in predicting income categories.

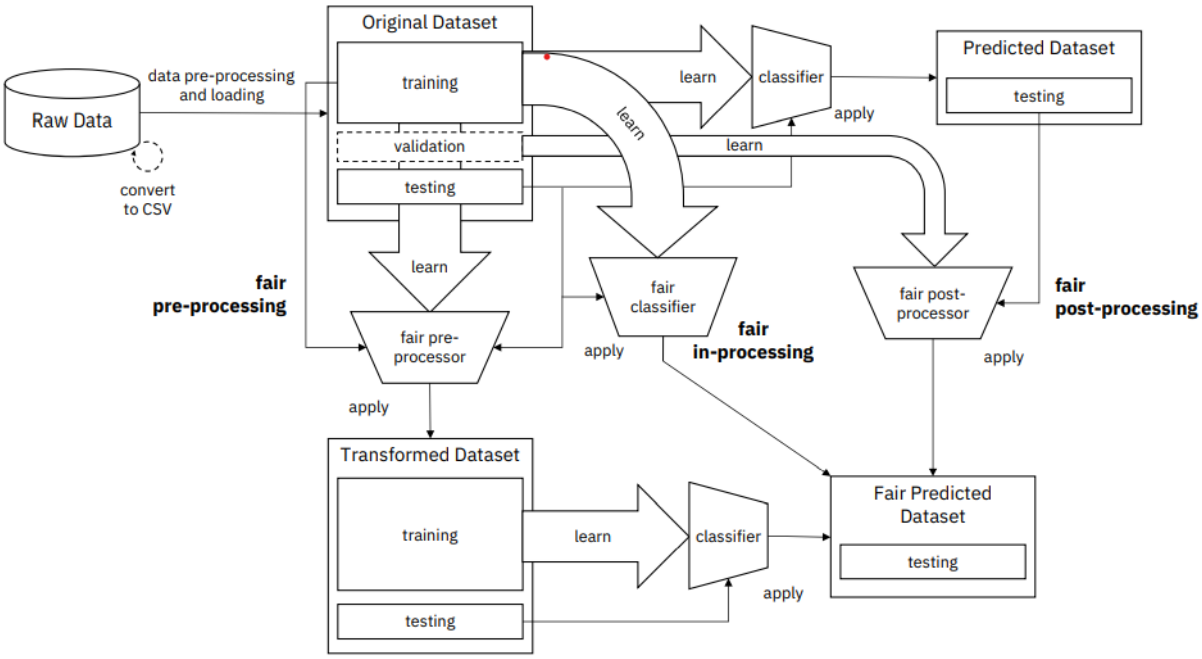

Figure 2: Schematic Design for the Fairness Pipeline

3.1 One-hot Encoding

The most common types of variables in structured data are continuous and discrete. Real numbers and integers can represent them. For example, if we use number 1 to represent red, number 2 to represent blue, and number 3 to represent yellow, we cannot assume that the size of the blue is more significant than the size of the red. We will map discrete values into a multi-dimensional space (Cerda, Varoquaux and Kégl, 2018). An example is shown in Table 1.

Table 1: Example of One-hot Encoding

Color	Colour_Red	Colour_Blue	Colour_Yellow
Red	1	0	0
Blue	0	1	0
Yellow	0	0	1

3.2 Normalization

A normalization concept can have various meanings. In statistics, it can be used to adjust values that are typically considered to be standard scales. The goal is to provide a uniform representation of normalized values across different datasets. Some types of normalization involve rescaling, which can arrive at values relative to a size variable. The normalization performed in the experiment is the standard score known as Z-score (Freedman, Pisani and Purves, 2007).

$$Standard\ Score(z) = \frac{x - \mu}{\sigma}$$

$$where: x\ is\ the\ data\ point,$$
$$\mu\ is\ the\ mean\ of\ the\ population,$$
$$and\ \sigma\ is\ the\ standard\ deviation\ of\ the\ sample$$

3.3 Class Imbalance

Imbalanced classification is a process that involves the representation and extraction of data that are not equal. This procedure is known as imbalanced learning. In terms of class imbalance, it refers to the issue of having more than two classes. Class imbalance means that datasets with multiple classes may have multiple minority classes. Class imbalance is a class issue that affects datasets that have more than two classes. A chief problem with machine learning datasets is that they tend to underperform well compared to standard ones. For instance, many algorithms that use class distribution data tend to make predictions when detecting multiple examples in each class. It is essential to learn that the minority class does not have as much importance as the majority class in machine learning. Balancing the classes will help machines perform better in the majority class (Fernández et al., 2018).

3.3.1 Balancing dataset with equal number of samples

Techniques designed to change the composition of a training dataset are commonly referred to as resampling methods. Sampling methods are so standard because they are straightforward to implement and can transform training datasets. The Naive Bayes Classifier learns the likelihood of observing a class from a given dataset. It constructs an artificial class distribution based on the input variable's specifics. Sample from the data set is used for training the model. It is not performed on the validation or holdout test datasets. Data sampling methods are commonly used to

address the issue of relative class imbalance in a training dataset. They ignore the underlying cause of the problem (He and Ma, 2013).

3.3.2 Balancing data set with equal number of ratios

In this method, we have balanced the dataset using the equal ratio of class compared to using the equal number of samples. Using this balancing technique to train the predictive models, the model can generalize well.

3.3.3 Balancing data set using data augmentation

In this method, we have balanced the data using the data augmentation technique. Data augmentation creates sample data from the data set is currently used. It is better to have more data as our classification model will have various data to learn from. In this example, we added a generated data point y_i for every data point x_i with a given gender and added it to our dataset. This technique increases the amount and variation of data in our data set.

3.4 Models/Algorithms

A machine learning model is expected to perform accurately on new tasks and scenarios after being trained on previously unseen data sets (Bishop, 2016). This goal is achieved by building a general model that can predict the probability of new events and scenarios. Computational learning theory is a branch of computer science that studies the performance of machine learning algorithms. It does not provide guarantees of the correctness of the algorithms. The complexity of the hypothesis must match the function underlying it. If the function is less complex than expected, the model will under-fitting the data. In computational learning theory, it is assumed that a computation can be performed in polynomial time if it can be done in terms of learning. However, it is not feasible to learn certain classes in polynomial time (Alpaydin, 2014). Different machine learning models exhibit different levels of bias. This project tries to identify less biased models by training different model types and architectures. Figure 3 gives an overview of the various problem groups and their corresponding algorithm family.

Overview of the problem group and its corresponding algorithm family

Defining the problem and algorithm family is an important stage of any machine learning and data science project. In this section, an overview of the problem group and its corresponding algorithm family is presented.

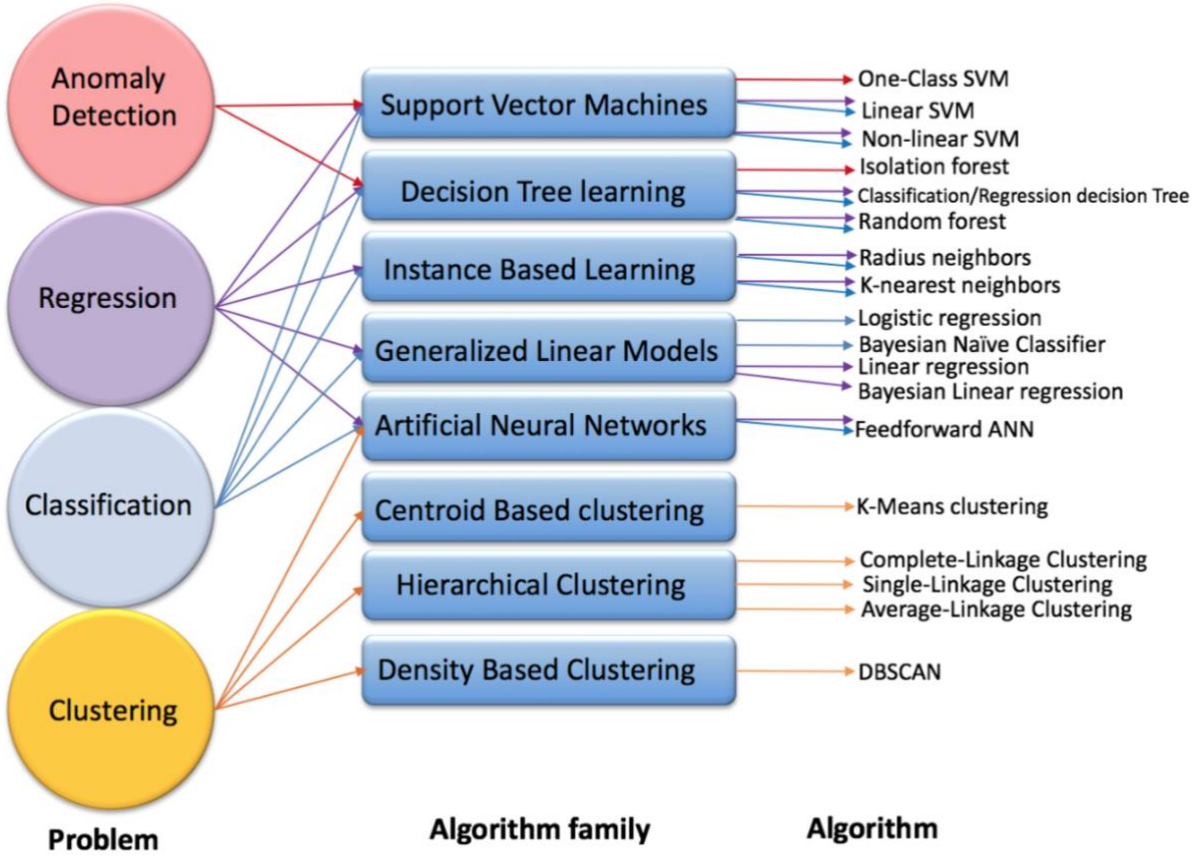

Figure 3: Overview of problem groups and its algorithm family

3.4.1 Support Vector Classifier

A training algorithm that uses SVM builds a model that takes a set of training examples and assigns new examples to one or the other. It is not a binary linear classifier. SVC is used to map the space between the two categories; new examples are then mapped into the same space and predicted to fall into one of the categories. Aside from linear classification, SVMs can also perform non-linear classification by mapping their inputs to high-dimensional feature spaces. Unsupervised learning is not possible when data is unlabeled. This method finds natural clustering within the data and then maps the resulting groups (Hastie, Tibshirani and Friedman, 2008). Representation of the Linear Support Vector Classifier is shown in Figure 4.

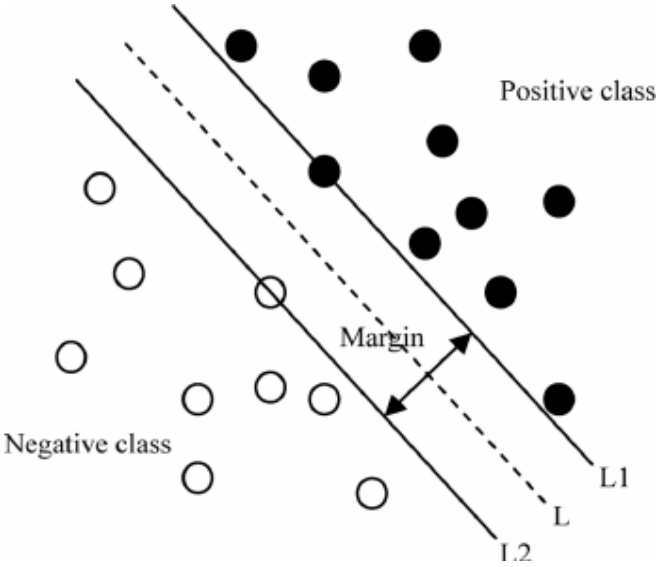

Figure 4: Representation of Linear Support Vector Classifier

3.4.2 Random Forest Classifier

Random forests are learning systems composed of several decision trees chosen by the trees for a given task. The outputs of the trees are then returned to the training set. For classifications, the output of random forests is the class that most trees have selected. They generally outperform gradient trees (Piryonesi and El-Diraby, 2020). While random forests can outperform decision trees in terms of accuracy, they do not provide the intrinsic interpretability that decision trees do. This ability to interpret data allows decision trees to make informed decisions. It lets developers confirm that the model has learned to make informed decisions (Hastie, Tibshirani and Friedman, 2008). Figure 5 shows the representation of the Random Forest Classifier.

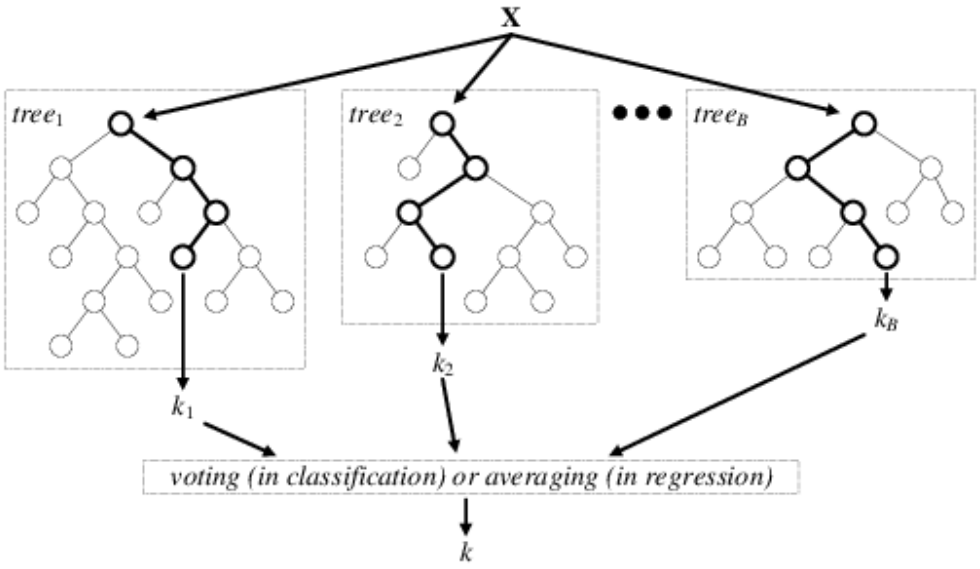

Figure 5: Representation of Random Forest Classifier

3.4.3 K-Nearest Neighbour Classifier

K-NN is a type of classification algorithm that only approximates a function locally. It avoids generating and processing all its data until it gets a complete evaluation. This algorithm relies on distance classification to determine the distance between objects and their respective physical units. If the objects have varying features, it can improve their accuracy significantly. A valuable technique for regression and classification is to assign weights to each of the neighbors' contributions. The k-nearest neighbour algorithm is sensitive to the structure of the data. It can be improved through supervised metric learning (Hastie, Tibshirani and Friedman, 2008). Representation of K-Nearest Neighbour Classifier is shown in Figure 6.

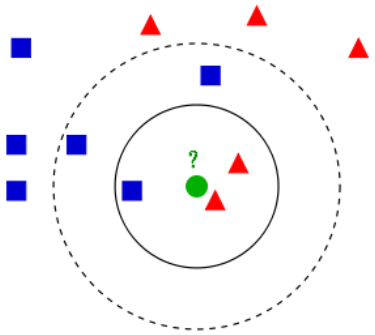

Figure 6: Representation of K-Nearest Neighbour Classifier

3.4.4 Logistic Regression Classifier

The statistical model is commonly used to model the probability of events in a particular class or event. Logistic regression can model events such as determining if an image contains a cat or a dog. A binary logistic regression model has two levels (categorical). If the dependent variable has more than two values, the output is modelled by multinomial ordinal logistic regression. The logistic regression model is a type of statistical model that does not perform statistical classifications. This is because it assumes that the output is likely to be uniformly distributed. For instance, if the input has a probability more significant than the cut-off, then classify it as one class below the other. The result is a binary classifier that uses the coefficients (Walker and Duncan, 1967). Figure 7 shows the schematic of a logistic regression classifier.

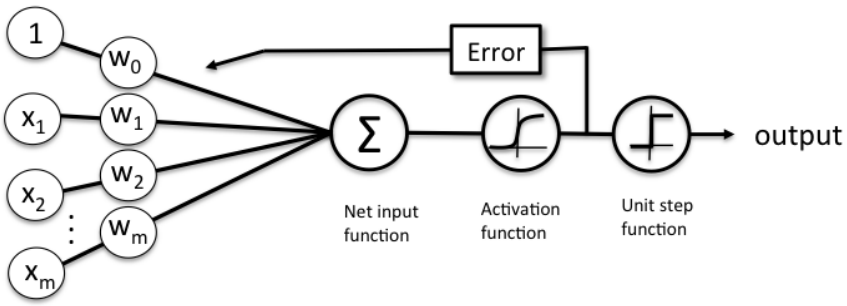

Figure 7: Schematic of a logistic regression classifier

3.4.5 Multi-Layer Perceptron Classifier

As shown in Figure 8, the multi-layer perceptron is an artificial neural network that maps sets of data to predefined output sets. It consists of multiple layers that are fully connected. The nodes of the layers are composed of neurons that use non-linear activation functions. There can be a non-linear hidden layer between the outputs and the inputs (Wasserman and Schwartz, 1988).

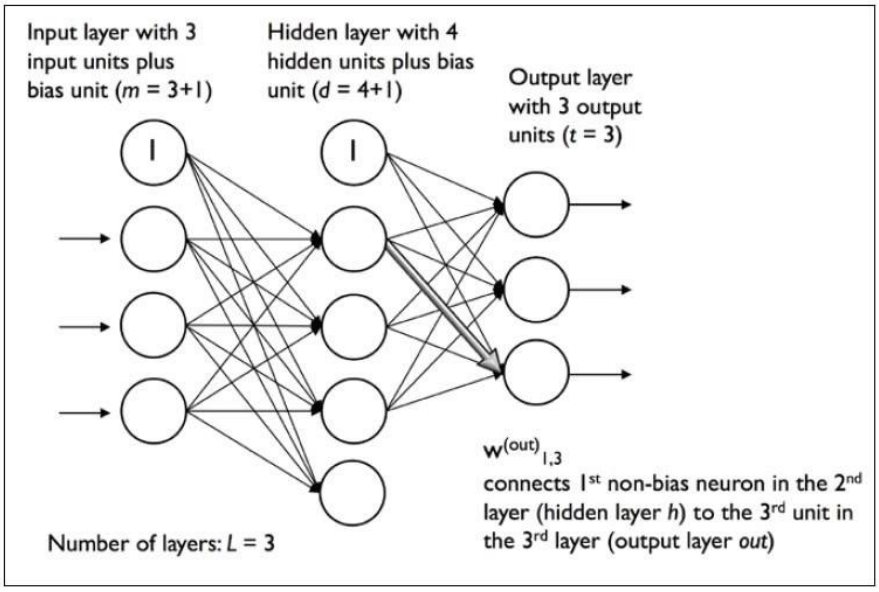

Figure 8: Schematic of Multi-Layer Perceptron Classifier

3.4.6 Multi Model Architecture

Stacking multiple classification methods can help improve performance over individual models. The goal of an ensemble modelling system is to improve its predictions over time. There are models and models architectures that contain components of ensemble learning methods. An ensemble learning technique is a process where multiple models are used to learn, as shown in Figure 9. The problem is that this method may not combine their predictions in the usual way (Xiao, Wu, Lin and Zhao, 2018).

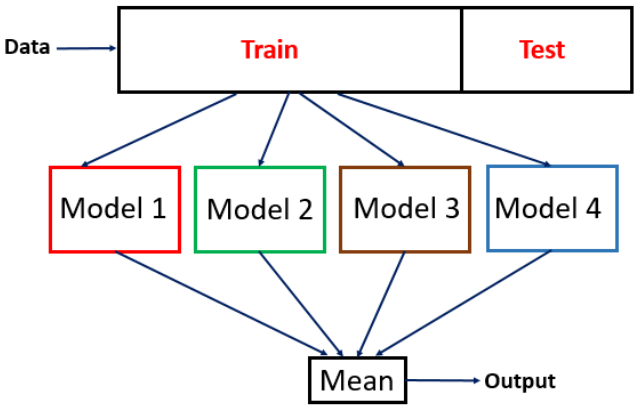

Figure 9: Multi Model Architecture

3.5 Bias Metrics

The debate about bias in machine learning has been around for a long time. Stories of how models go wrong have made headlines, and journalists and humanitarian lawyers have also contributed to discussing what values are represented in these models. While human bias is an issue that's not always easily defined, it is still essential to know how to identify it in machine learning models. There are many tests that we can perform to test different types of bias in our model. The goal is to make it easier to communicate and to mitigate bias. Identifying bias is the first step in the process of mitigating it in our model. The right combination of tests and mitigation techniques can help us improve our machine learning model and minimize bias. This section will understand the bias metrics such as statistical parity, disparate impact, and equality of opportunity. Figure 10 represents the metrics used to measure bias. Table 2 lists out the metrics used to measure bias.

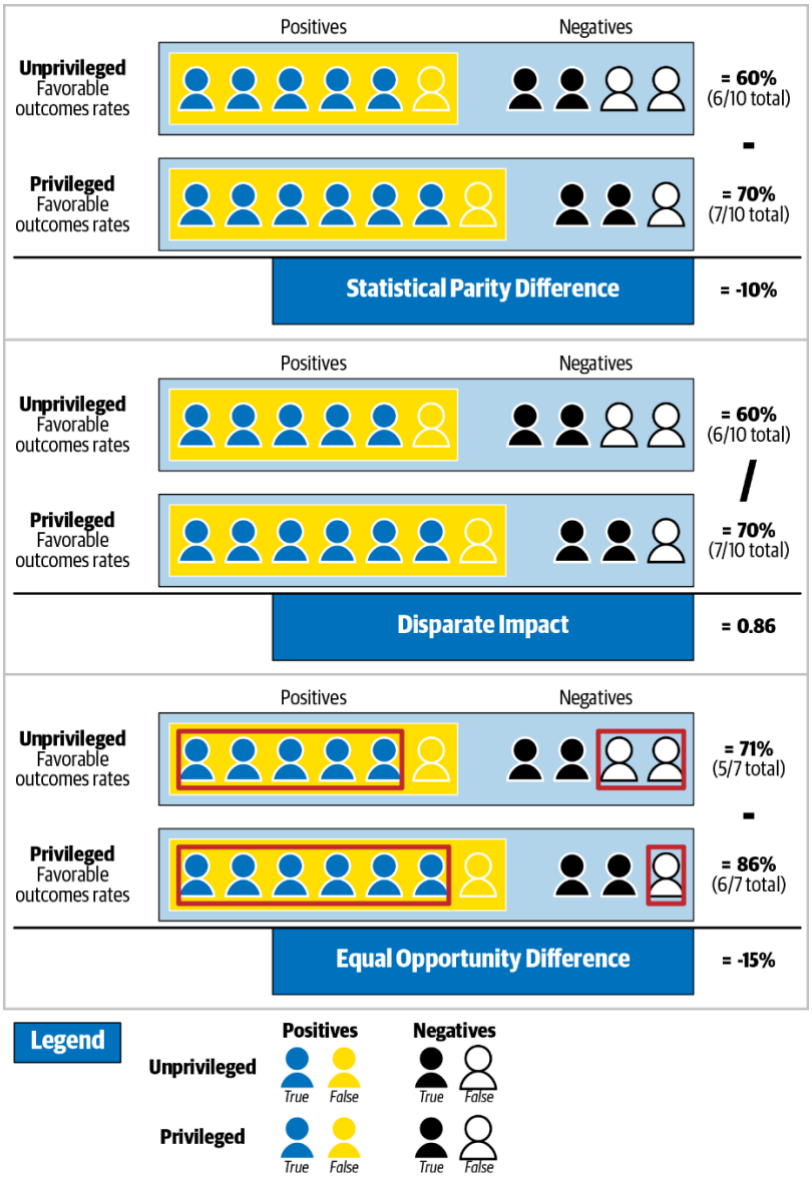

Figure 10: Representation of bias metrics

3.5.1 Statistical Parity

Statistical parity is a test that measures the bias against different groups of people based on the protected attribute. In statistical parity, the goal is to measure if the various groups have equal chances of achieving a favourable outcome. This concept is usually used for a hiring model. Figure 11 will show how to implement statistical parity tests to analyze and confirm the pre-defined data sets (Verma and Rubin, 2018).

$$Pr\left[h(x) = 1 | x \in P^C\right] = Pr\left[h(x) = 1 | x \in P\right]$$

Figure 11: Equation for Statistical Parity

3.5.2 Disparate Impact

Disparate impact is a statistical concept that takes the probability of positive outcomes for the unprivileged groups instead of the privileged ones. Disparate impact concerns issues arise when policies and procedures are used in various ways to affect a class of people based on factors such as race, colour, and religion. For instance, it is common for employers to require specific characteristics of people for certain jobs. Today, AI technology is being used to make various decisions in almost every aspect of our lives. They can help us make mundane tasks more manageable, predict our credit rating, or even provide us with life-changing advice. The calculation divides the positive outcome by the number of unprivileged people who received it (Feldman et al., 2015). The equation for Disparate Impact is displayed in Figure 12.

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

Figure 12: Equation for Disparate Impact

3.5.3 Equality of opportunity

Equality of opportunity checks that a preferred label is equally well-suited to a given attribute. Equality of opportunity is a concept that refers to the likelihood that a person will be able to participate in an equal opportunity regardless of their group membership status. equality of opportunity is not only concerned with the true positive but also concerns the unprivileged groups. Let X be the set of features that a model can use to input data, and A be the protected attribute. For equal opportunity, we need to agree that the odds are equal in the case that Y=0, where Y is a binary decision, as shown in Figure 13 (Hardt, Price and Srebro, 2016).

$$P\{\hat{Y} = 1 | A = 0, Y = 1\} = P\{\hat{Y} = 1 | A = 1, Y = 1\}$$

Figure 13: Equation for Equality of Opportunity

Table 2: List of metrics used to measure bias

Metric	Description	Interpretation
Accuracy	Model performance, percentage of correct predictions.	A decimal value between 0 and 1 with one correctly predicting all test cases.
Statistical Parity Difference	Difference of the rate of favourable outcomes received by an underprivileged group to the privileged group.	A value between -1 and +1, negative values favour privileged groups while positive values favour unprivileged groups.
Equal Opportunity Difference	The difference of true positive rates between unprivileged and privileged groups.	A value between -1 and +1, negative values favour privileged groups while positive values favour unprivileged groups.
Disparate Impact	The ratio of the rate of a favourable outcome for the unprivileged group compared to the privileged group.	The ideal value of 1 implies perfect fairness. A value less than one favours privileged group while greater than 1 favours unprivileged group.

3.6 Evaluation Metrics

Statistics is an essential component of a machine learning project's development. It is used to evaluate the skills of a machine learning model. The accuracy of a machine learning algorithm is a way to measure how often it classifies a data point correctly. A true positive (TP) or a true negative (TN) is a data point that an algorithm correctly classified. A false positive (FP) or a false negative (FN) is a data point that an algorithm classified incorrectly. Various metrics use different ratios of TP, TN, FP, FP to evaluate the algorithm's performance to classify data points. Data points present in the data set are known as actual values, and the data points predicted by the model are known as predicted values. Figure 14 represents a confusion matrix used in evaluating the predictions made by the machine learning models. A list of the evaluation metrics is given in Table 3. An example of the ROC and AUC are illustrated in Figure 15.

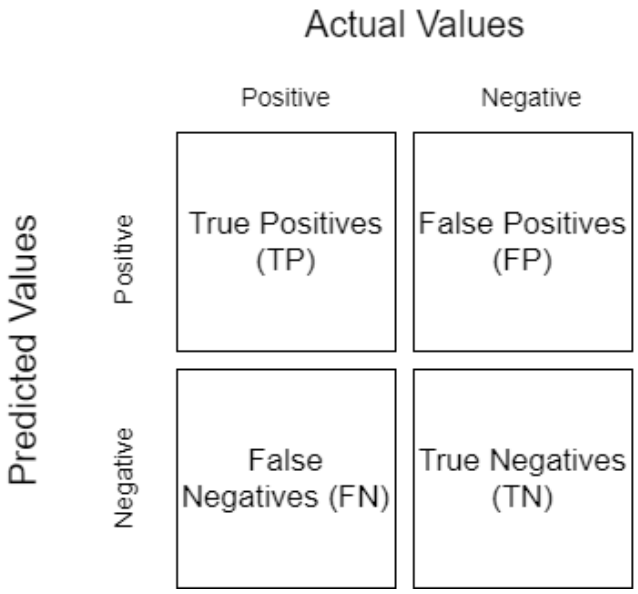

Figure 14: Diagram of Confusion Matrix

Table 3: List of evaluation metrics

KPI	Formula	Explanation
Accuracy	$\frac{(TP+TN)}{(TP+FP+TN+FN)}$	How good the model is at predicting the correct data point
Positive rate	$\frac{TP}{(TP+FP)}$	The ratio of the number of true positives and the total positives
Negative rate	$\frac{TN}{(TN+FN)}$	The ratio of the number of true negatives and the total negatives
True positive rate (TPR)	$\frac{TP}{(TP+FN)}$	The probability that an actual positive will test positive
True negative rate (TNR)	$\frac{FP}{(FP+TN)}$	The probability that an actual negative will test negative
Receiver operating Characteristic (ROC)	$TPR\ vs\ FPR$	The ROC curve plots the true positive rate against the false positive rate.
Area Under ROC Curve (AUC)	$TPR\ vs\ FPR$	Performance metric that shows how well a model can identify classes at various threshold settings.

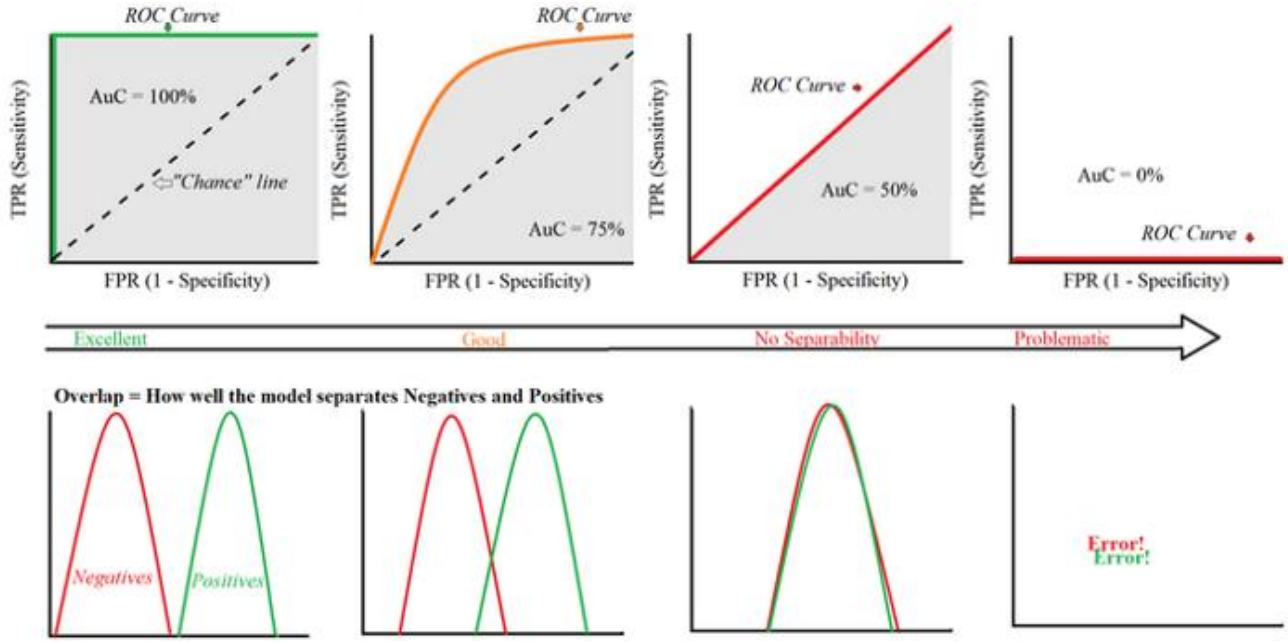

Figure 15: Example of ROC and AuC curves

4.0 Experimentation and Results

4.1 Data Collection

This project was carried out on the UCI Adult dataset by establishing familiarity with the dataset and explore income distributions across different demographics in the dataset. The UCI adult dataset is one of the most polar machine learning datasets. It is publicly available on the internet on the UCI machine learning repository. The dataset comprises more than 48,000 data points extracted from the 1994 census database in the United States. Each data point in the dataset is comprised of 15 features. These include age, work class, education, relationship, race, sex, salary, etc., as listed in Table 4.

Table 4: Data Description

Features	Description
Age	Age of an individual
Workclass	Individuals' work category
Final-Weight	Weight estimates assigned by the census bureau
Education	Individuals' highest education degree
Education-number-of-years	Individuals number of years spent in education
Marital-status	Individuals' marital status
Occupation	Individuals' occupation
Relationship	Individuals' relation in a family
Ethnicity	Race of an individual
Sex	Gender as per birth
Capital-gain	Gain per capital
Capital-loss	Loss per capital
Hours-per-week	Individuals' working hours per week
Native-country	Individuals' native country
Salary	Individuals' salary

4.2 Data Understanding

Figure 16 shows us the gender distribution in the dataset. We can observe that about 16,000 individuals identify as female. Furthermore, about 32,000 individuals identify as male.

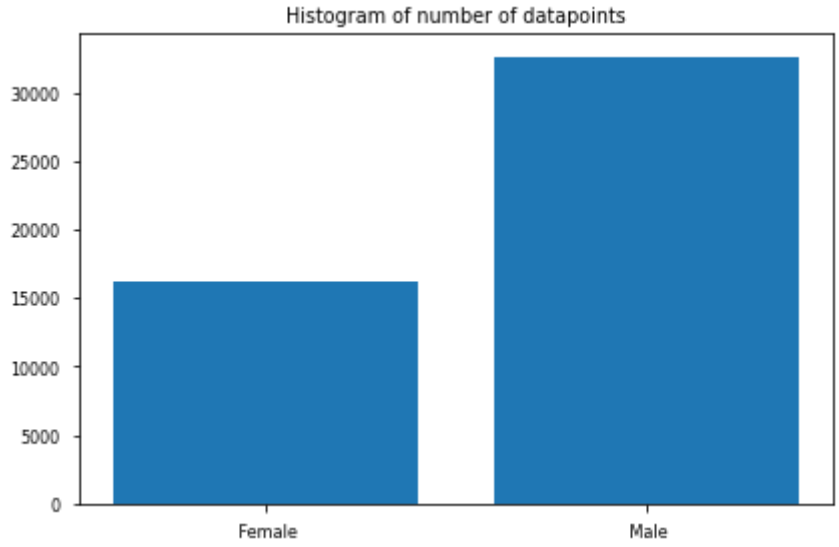

Figure 16: Histogram of males and females in the census

Looking at the race distribution in Figure 17, we can observe that slightly more than 40,000 individuals identify as white. Furthermore, about 4,000 to 5,000 individuals identify as black. The rest are other minorities.

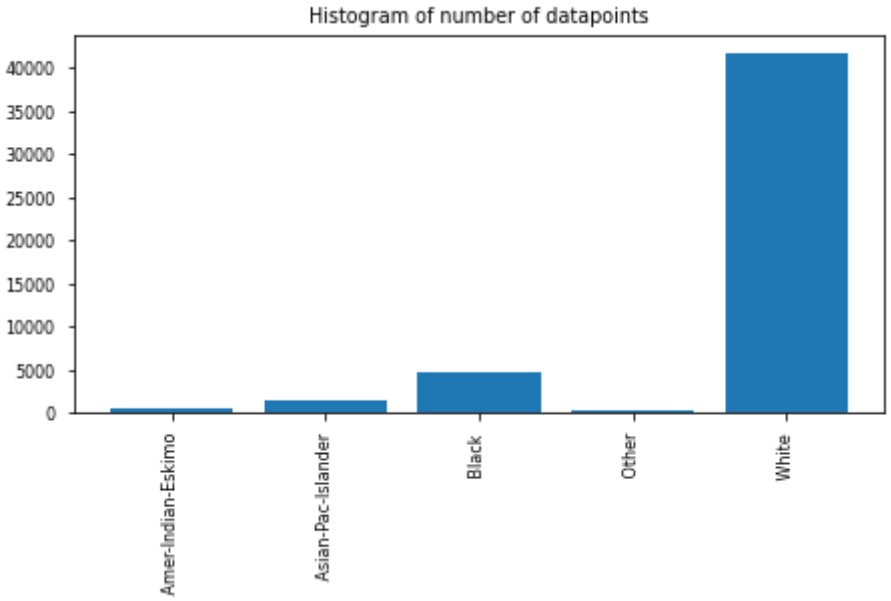

Figure 17: Histogram of ethnicities in the census

If we look at the native country distribution in Figure 18, we can observe that most individuals in the data set are from the United States.

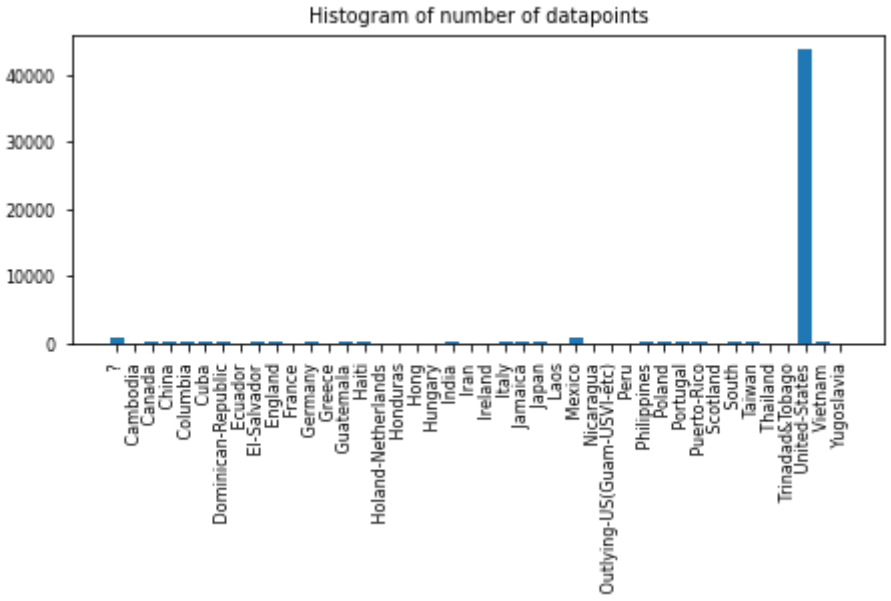

Figure 18: Histogram of native countries in the census

If we look at the education distribution in Figure 19, we can observe that number of high school graduates is more than 33 per cent compared to college graduates and almost 50 per cent more compared to individuals with bachelor's degrees.

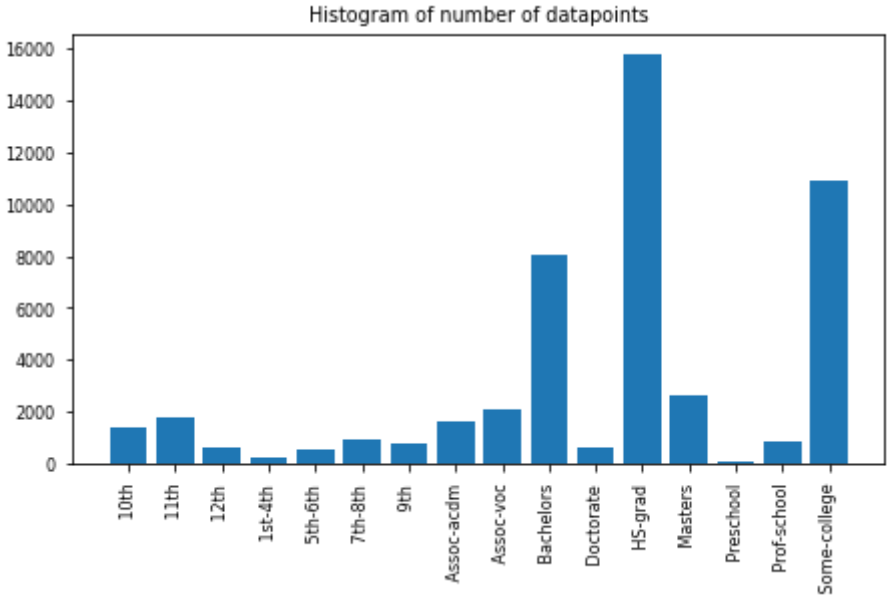

Figure 19: Histogram of education distribution in the census

If we look at the occupation distribution in Figure 20, we can observe an unequal representation of the occupation groups in the dataset.

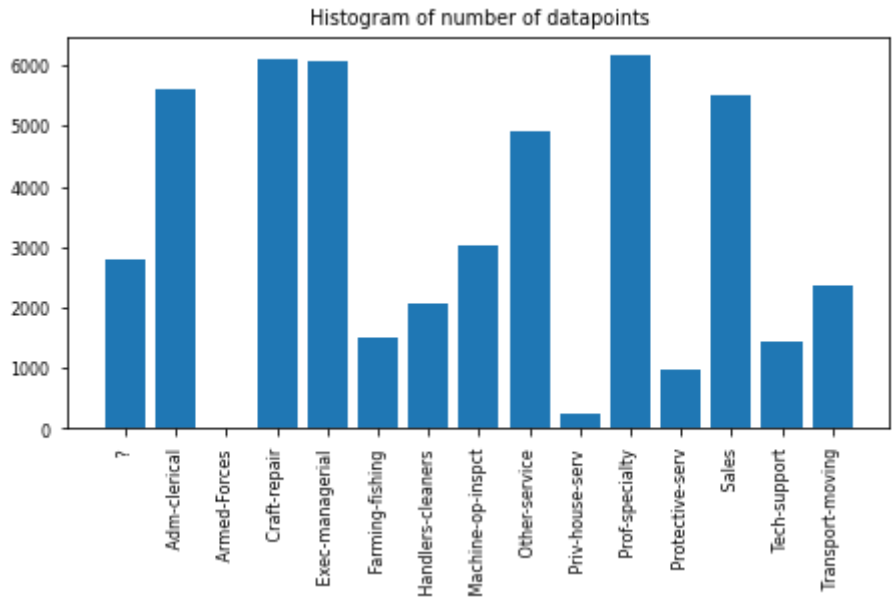

Figure 20: Histogram of occupation in the census

If we look at the distribution of the income category in the general population as illustrated in Figure 21, we can observe that about 37,000 individuals earn less or equal to $50,000. Furthermore, only about 12,000 and 13,000 individuals earn more than $50,000.

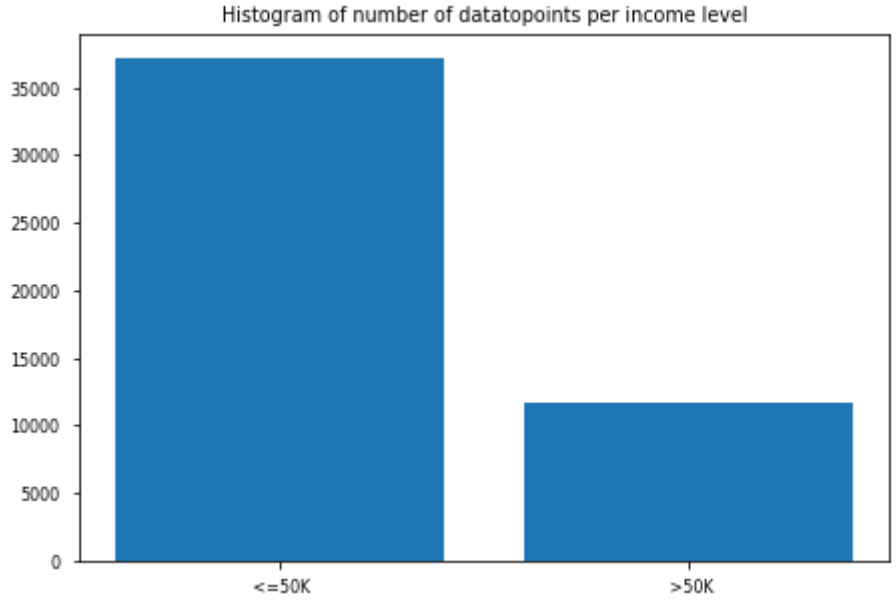

Figure 21: Histogram of number of datapoints per income level

By looking at the income distribution across various levels as illustrated in Figure 22, it can be observed that the male demographic's ratio of making more than $50,000 is about a third, while the female demographic's ratio is about 20%.

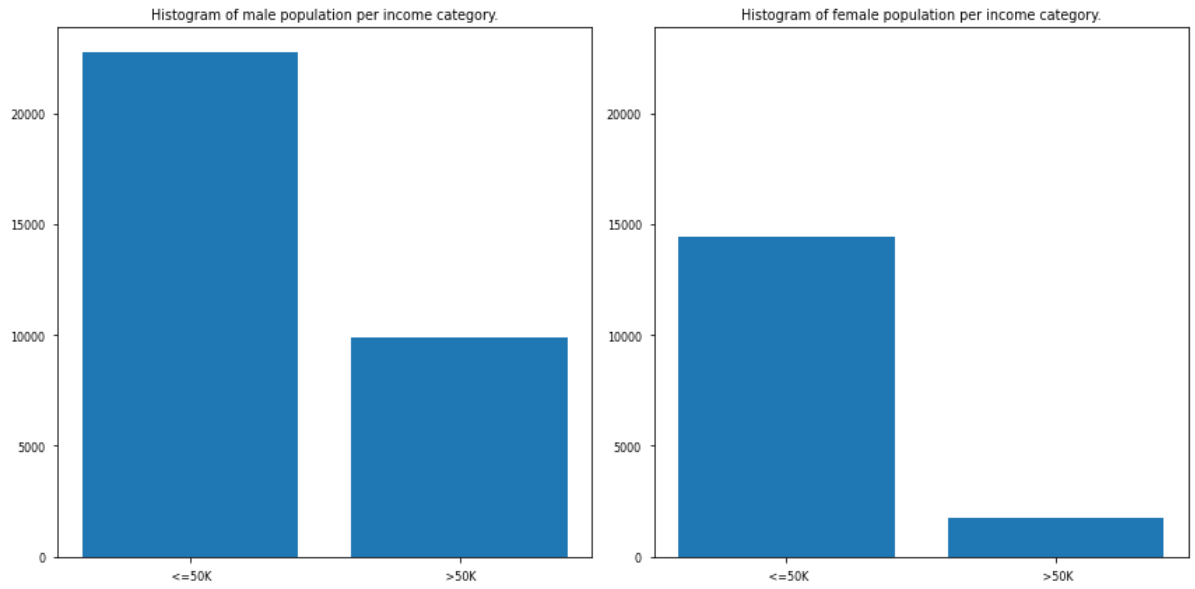

Figure 22: Histogram of gender per income category

Looking at the distribution of the working hours per week in Figure 23, we can observe that most individuals spend about 30 to 40 hours per week working.

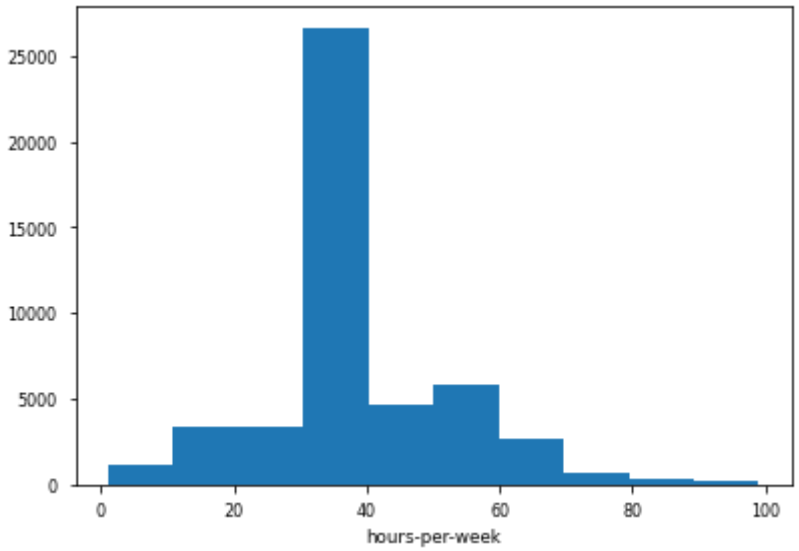

Figure 23: Histogram of hours-per-week in the census

An important observation from the number of data points that the male population has is significantly higher than that of the female population. Therefore, it is imperative to think about how this representation disparity might affect predictions of a model trained from this data.

4.3 Data Preparation

This section of the project explored the steps involved in transforming raw data into an appropriate categorical or numerical representation for machine learning tasks.

i. The process began by creating a copy of the original data and drop the missing values present.
ii. In this experiment, we have decided to assign a binary label '0' to individuals from the United States and another binary label '1' to individuals outside the United States, which is illustrated in Table 5.

Table 5: Transforming native-country attribute using binary encoding

Attribute	Binary Label
United States	0
Non-United States	1

iii. We have applied the same transformation to the 'sex' and 'salary' attribute since each one of these attributes has two possible outcomes, therefore making binary representation appropriate illustrated in Tables 6 and 7.

Table 6: Transforming gender using binary encoding

Attribute	Binary label
Male	1
Female	0

Table 7: Transforming salary using binary encoding

Attribute	Binary label
<=50K	1
>50K	0

iv. Since categorical features can take multiple values, we typically use one-hot encoding for specific attributes. However, this is often a decision that must be made on a case-by-case basis, depending upon the application.
v. In addition, converting work classes to binary features is not ideal if the people from different categories have varying income levels.
vi. Finally, we convert the capital gain, and capital loss attributes to binary.

4.4 Illustrating Gender Bias

In this part of the experiment, we will demonstrate the gender bias in our data. We will then apply the same machine learning approach to estimate income categories without mitigating bias.

We start with normalizing the continuous variables such as age, final weight, education number and hours per week. Then split the data set into training data set and test data set. We then feed the Multi-layer Perceptron (MLP) classifier on the training data and then use the model to predict the test data.

Multi-Layer Perceptron (MLP) model belongs to the class of feedforward neural networks. Each node uses a non-linear activation function, giving the model ability to separate non-linear data. Multi-Layer Perceptron (MLP) is trained using the backpropagation technique. However, a few downsides are that the model suffers from overfitting, which is not easily interpretable.

Before we evaluate our model, let us start by establishing the terminology. The high-income category is the group of individuals who earn more than $50,000 a year. Furthermore, the low-income category is the one that includes those who earn less than $50,000 a year.

Let us look at different error rate metrics for the model that we trained previously across gender demographics, as illustrated in Figure 25. If we look at accuracy, we can observe that the accuracy for the male demographic is about 80% or 0.8, while the accuracy for the female demographic is about 90% or 0.9. It can be observed that the male demographics have higher positive and true positive rates when compared to the female demographic. Moreover, the female demographic has higher negative and true negative rates compared to the male demographic.

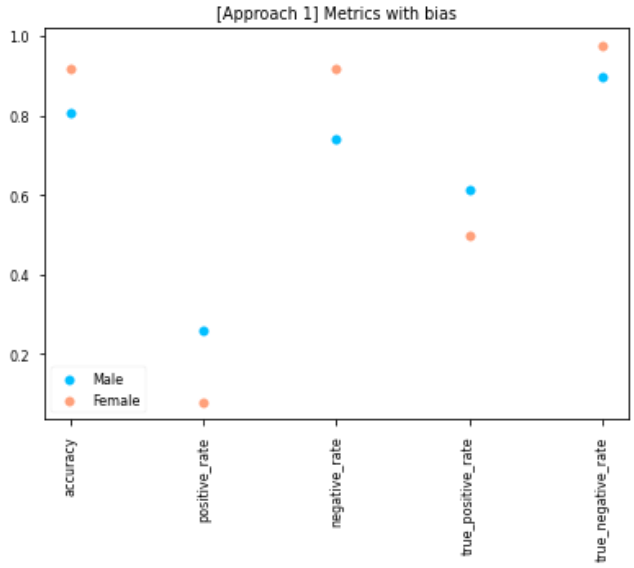

Figure 25: [Approach 1] Metrics with bias

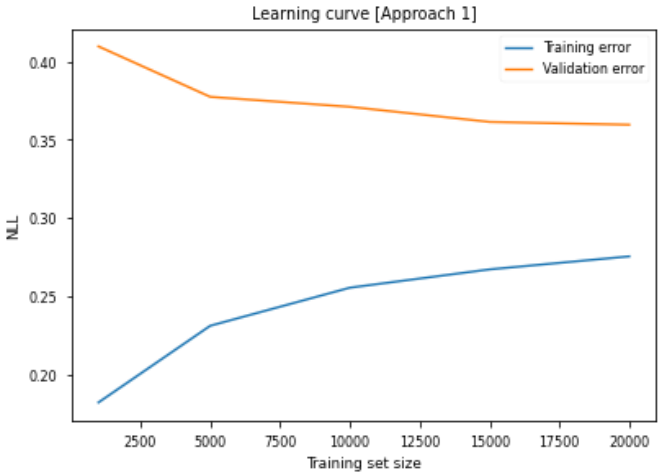

Figure 24: [Approach 1] Learning curve with bias

4.5 Data-based Debiasing Techniques

In this section, we will be exploring three primary techniques for reducing bias in machine learning. The first technique is called unawareness, the second technique is database re-balancing, and the third technique is data augmentation. The motivation behind this is our hypothesis that gender bias could come from the unequal representation of male and female demographics in our dataset. We, therefore, attempt to recalibrate and augment the dataset to equalize gender representation in our training data.

4.5.1 Mitigation through unawareness

The first data-based technique that we are going to explore is called debiasing by unawareness. This technique attempts to decrease algorithmic bias by removing sensitive/protected attributes from training data. Removal of sensitive/protected attributes is called unawareness. The results obtained from mitigating bias through unawareness, as shown in Figure 27, indicate that although we did not improve the accuracy of our calculations, nor did we not reduce the gap. We reduced the gap for other metrics, like the positive rate, the negative rate, the true positive rate, and the true negative rate. Moreover, this is an example of how debiasing techniques might not improve all the metrics. However, it might see a significant reduction in the gap for the other metrics of interest.

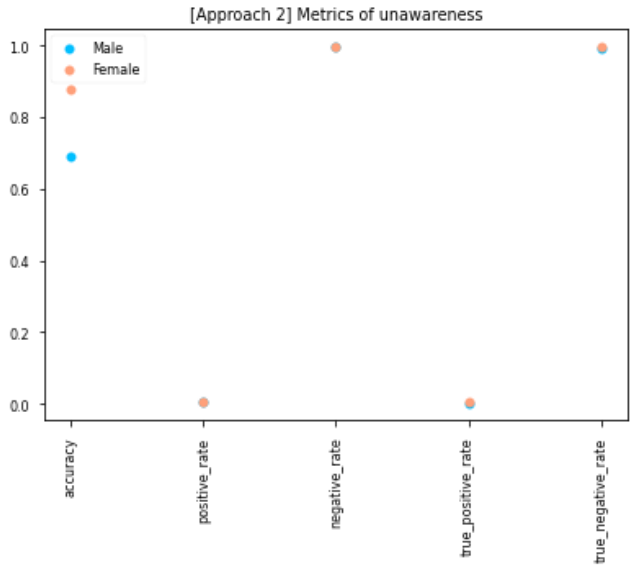

Figure 27: [Approach 2] Metrics of unawareness

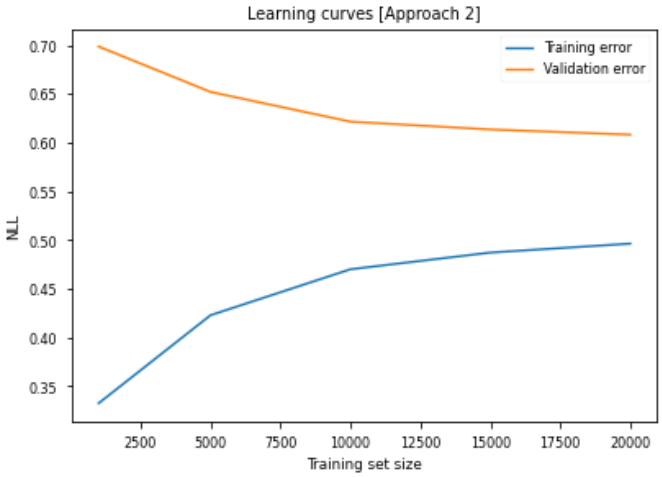

Figure 26: [Approach 2] Learning curve of unawareness

Mitigating bias by unawareness can help remove some biases, but it can also be ineffective if other features which are being dropped correlate with the protected attributes. These attributes are referred to as proxy variables (Pedreshi, Ruggieri and Turini, 2008).

4.5.2 Mitigation through data set balancing

The second data-based debiasing technique that we will explore is to equalize the representation by using the equal number of an equal ratio of male and female individuals in our dataset or within each income category. The steps carried out are listed below.

Equal number of data points per demographic

In this method, we start by attempting to equalize the number of data points per gender category. Moreover, in this approach, we will draw a sample with an equal number of data points from the male and the female demographics. We have

obtained these results as shown in Figures 29 and 31 from training the Multi-Layer Perceptron (MLP) Classifier on a data set with an equal number of data points per income level in each gender category.

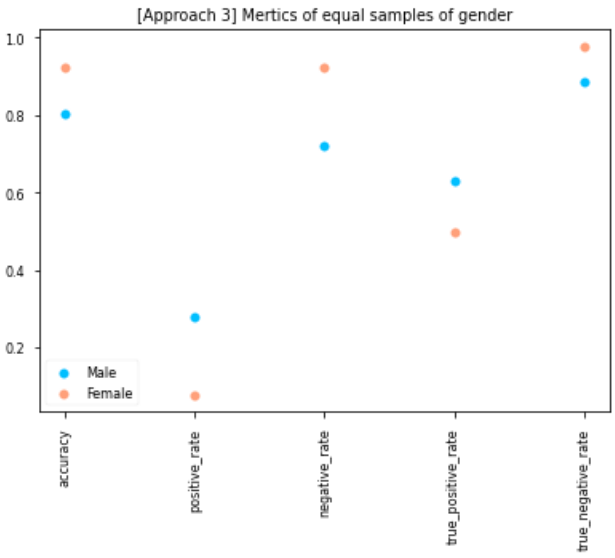

Figure 29: [Approach 3] Metrics of equal sample of gender

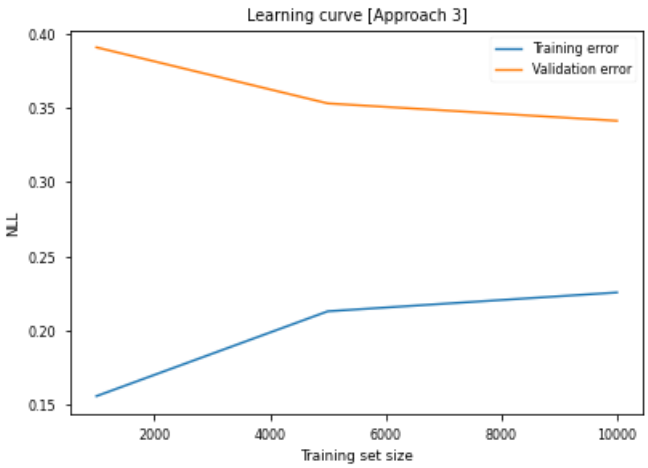

Figure 28: [Approach 3] Learning curve of equal sample of gender

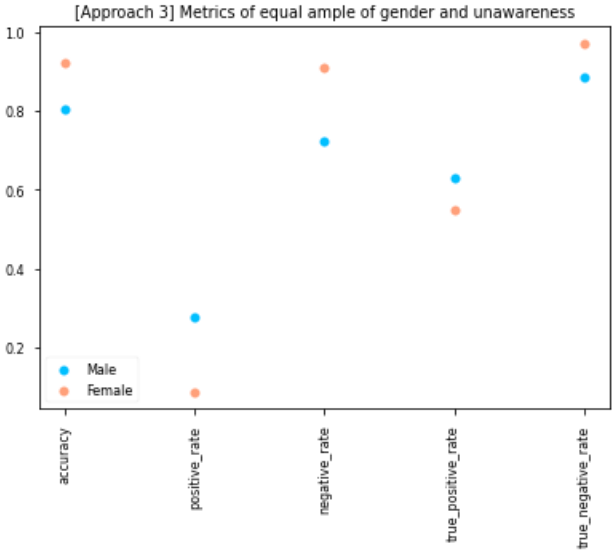

Figure 31: [Approach 3] Metrics of equal sample of gender and unawareness

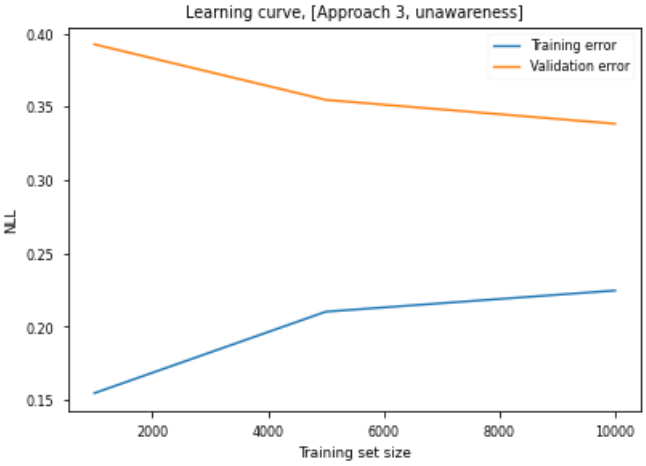

Figure 30: [Approach 3] Learning curve of equal sample of gender and unawareness

Equal number of data points per demographic in each category

The next attempt will equalize the number of data points per gender category. Furthermore, the number of high-income and lower-income earners is the same for the male and female demographic in the sample we use to train the model. Here are the key metrics from the model based on a data set sample that shows

an equal number of data points for each gender category illustrated in Figures 32 and 33.

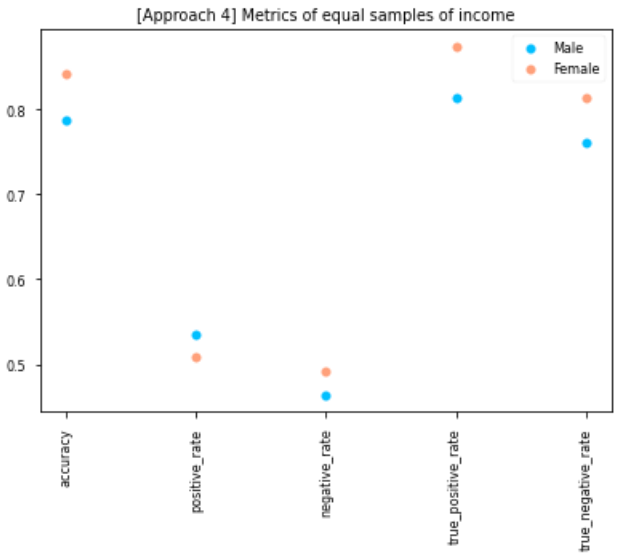

Figure 32: [Approach 4] Metrics of equal samples of income

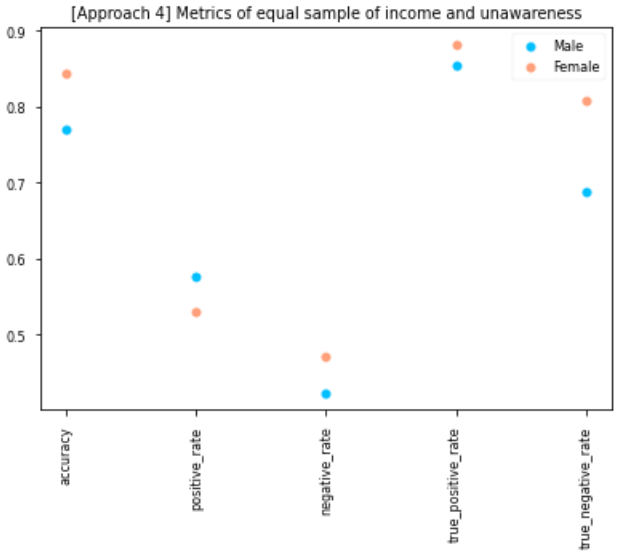

Figure 33: [Approach 4] Metrics of equal samples of income and unawareness

This method limits the number of data points that can be collected from each demographic. This method limits the data set's size depending on the smallest demographic. If the smallest demographic has a minimal number of data points, we will have a minimal training data set. Therefore, we can obtain a larger sample size by equalizing the ratio in some cases instead of equalizing the number of data points by demographic. Furthermore, that is what we are going to look at in the following approach.

Equal ratios instead of equal number of data points

This method is used to equalize the ratio of high-income and low-income levels in each category. The ratio of high-income and low-income levels is equalized to the ratio of the male demographic. Similarly, we do this for the female demographic as well. This methodology results in a higher sample size.

Following is the plot for the results of this methodology illustrated in Figures 35 and 37. We can observe that, although there is some gap between the accuracy and the true positive rate, the gap is way more minor for the positive rate, the negative rate, and the true negative rate.

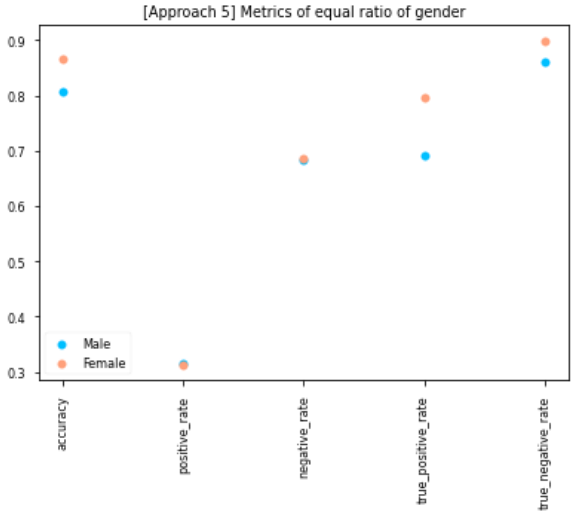

Figure 35: [Approach 5] Metrics of equal ratio of gender

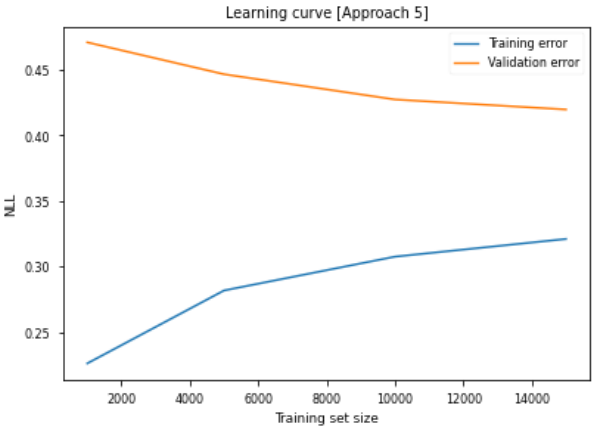

Figure 34: [Approach 5] Learning curve of equal ratio of gender

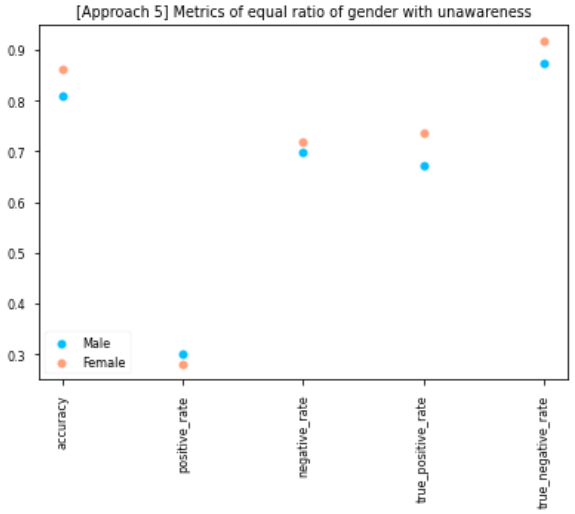

Figure 37: [Approach 5] Metrics of equal ratio of gender with unawareness

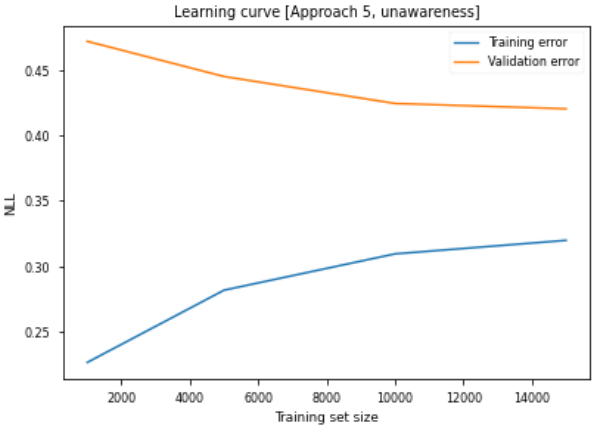

Figure 36: [Approach 5] Learning curve of equal ratio of gender with unawareness

4.5.3 Mitigation though data augmentation

The following technique we are going to look at is counterfactual augmentation. Furthermore, in this approach, for each data point x_i with the gender, we generate a new data point y_i that differs from x_i only at the gender attribute. Moreover, we add y_i to our training data set.

By looking at the results of augmentation in Figure 39 and 41, we can see that the gap between the male and the female demographic has pretty much disappeared. Moreover, this is what we want and expect from a fair machine learning model.

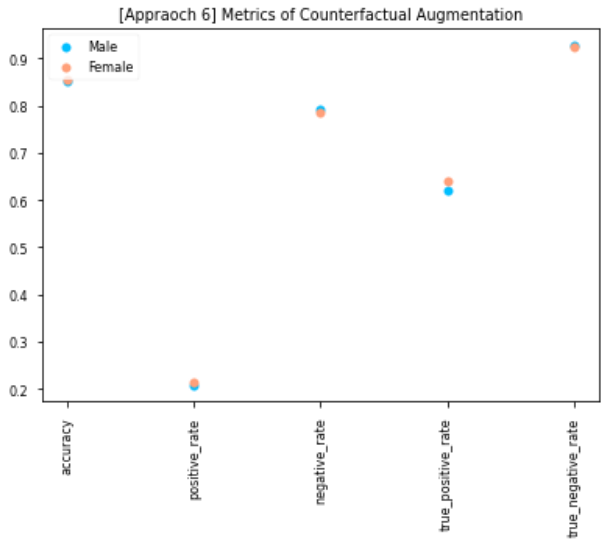

Figure 39: [Approach 6] Metrics of counterfactual augmentation

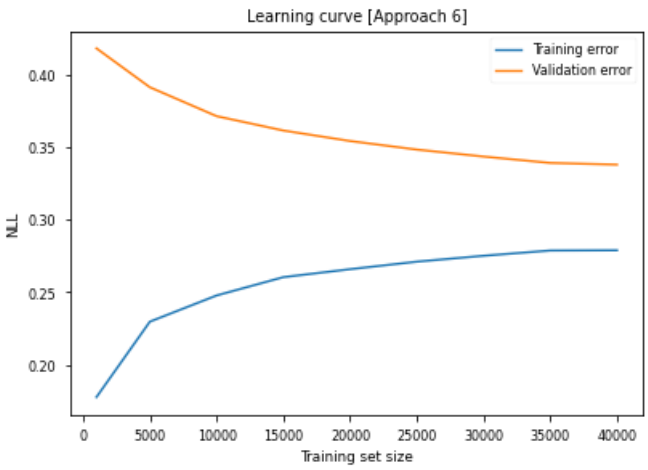

Figure 38: [Approach 6] Learning curve of counterfactual augmentation

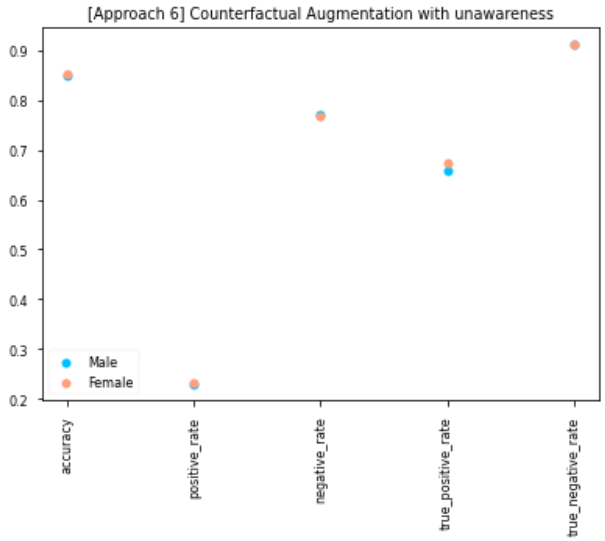

Figure 41: [Approach 6] Metrics of counterfactual augmentation with unawareness

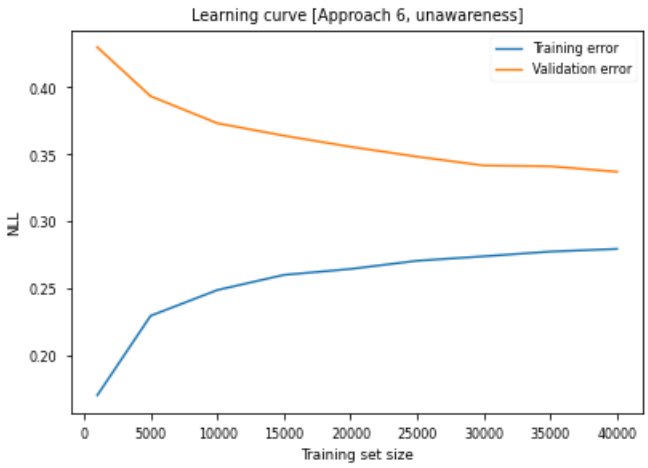

Figure 40: [Approach 6] Learning curve of counterfactual augmentation with unawareness

4.6 Model-based Debiasing Techniques

In this part of the experiment, we are going to explore model-based debiasing techniques. Moreover, we will look at different model types and architectures and examine how they perform for males versus females to determine the least biased model.

4.6.1 Debiasing through single-model architecture

The objective is to find out which model has the least amount of bias. This is done by changing the type of architecture or model type. We can observe which one tends to be inherently less biased. Moreover, these are the ones we will choose in our experiment since this is a classification problem listed in Table 8.

Table 8: List of problem groups and its corresponding algorithm family

Problem	Algorithm family	Algorithm
Anomaly Detection	Support Vector Machines	Non-linear SVM
Regression	Generalized Linear Models	Logistic Regression
Regression	Decision Tree Learning	Random Forest
Classification	Instance Based Learning	K-nearest Neighbors
Classification	Generalized Linear Models	Bayesian Naïve Classifier
Clustering	Artificial Neural Networks	Multi-layer Perceptron Classifier

For simplicity, we have used different parameters for different models. However, in a practical setting, we could use a technique like cross-validation or hyperparameter to find out the best parameter to use for each model. The resulting accuracy for the models is as follows. Results are listed in Table 9.

Table 9: Results of the single-model classifiers

Model	Accuracy
Logistic Regression	0.8405
Random Forest	0.8314
Gaussian Naïve Bayes	0.6405
Multi-Layer Perceptron Classifier	0.8406
Support Vector Classifier	0.8450
K-Nearest Neighbors	0.8272

4.6.2 Debiasing through multi-model architecture

We also examined multi-model architectures. A multi-model architecture combines different machine learning models and makes a prediction by considering the prediction of multiple models. There are different ways a multi-model architecture can make a decision, but here we will be looking at a Voting classifier that uses either

the Soft Voting rule or Majority/Hard Voting rule to make predictions. The hard voting consensus is the consensus among the various models. The soft voting consensus is the average prediction among the models. We use scikit-learn 'VotingClassifier' to combine single models and train them all at once. The results of the voting classifier are listed in Table 10.

Table 10: Results of the voting classifier

Voting Classifier	Accuracy
Hard voting accuracy	0.8443
Soft voting accuracy	0.8412

5.0 Evaluation

5.1 Bias mitigation approaches

This section of the report examines and evaluates all the metrics on all the approaches.

5.1.1 Comparing overall accuracies

In terms of overall accuracy, as shown in Figure 42, some techniques lead to higher degrees of accuracy, while others do not. For instance, counterfactual augmentation and class balancing are both techniques that can lead to higher accuracy. However, we can also observe in Figure 42 that some techniques like gender unawareness do not always guarantee higher accuracy.

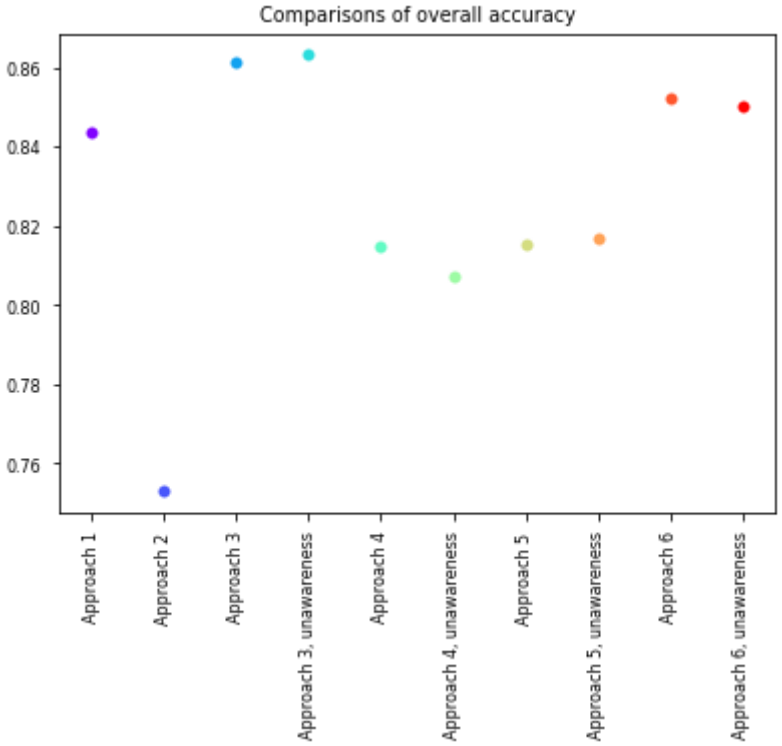

Figure 42: Comparison of overall accuracy

5.1.2 Comparing overall accuracy across gender

Looking at the accuracy across gender in Figure 43, we can observe that the counterfactual augmentation technique still has the smallest gap between males and females. However, we can also observe in Figure 43 that some techniques like gender unawareness still have a slightly higher accuracy gap in male demographics versus female demographics.

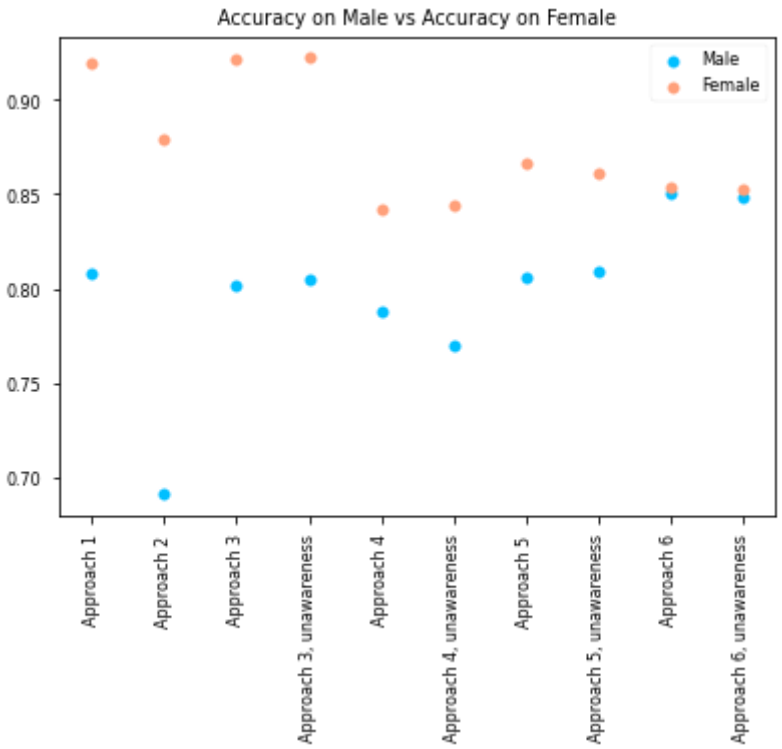

Figure 43: Comparing overall accuracy across gender

5.1.3 Comparing positive and negative rates across gender

The plots in Figures 44 and 45 show the comparison between the positive and negative rates for different techniques when it comes to gender attributes.

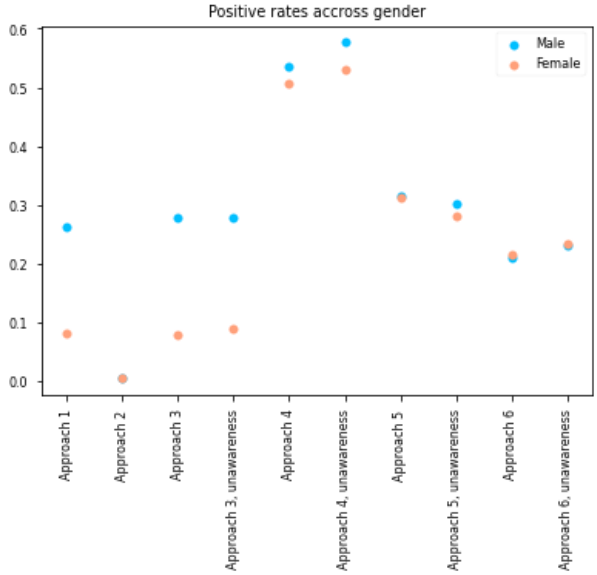

Figure 45: Comparing positive rates across gender

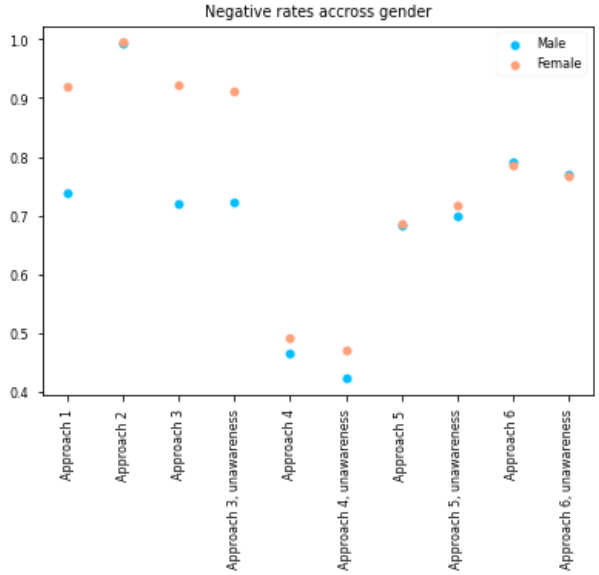

Figure 44: Comparing negative rates across gender

5.1.4 Comparing True positive and True negative across gender

The plots in Figures 46 and 47 show the relative rates of positive and negative rates across gender.

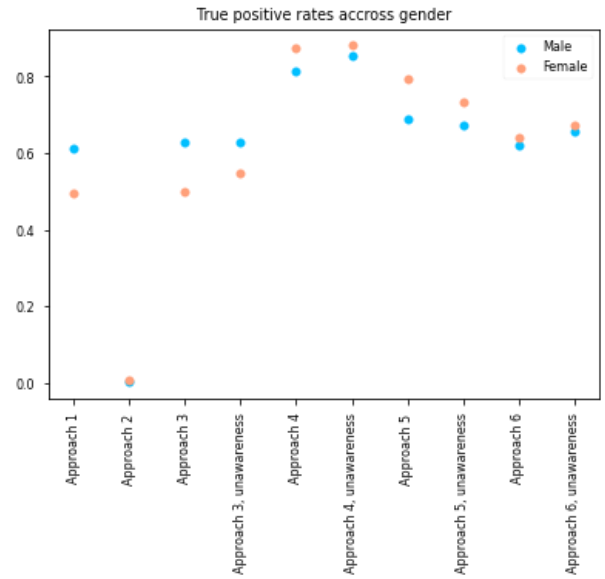

Figure 47: Comparing true positive rates across gender

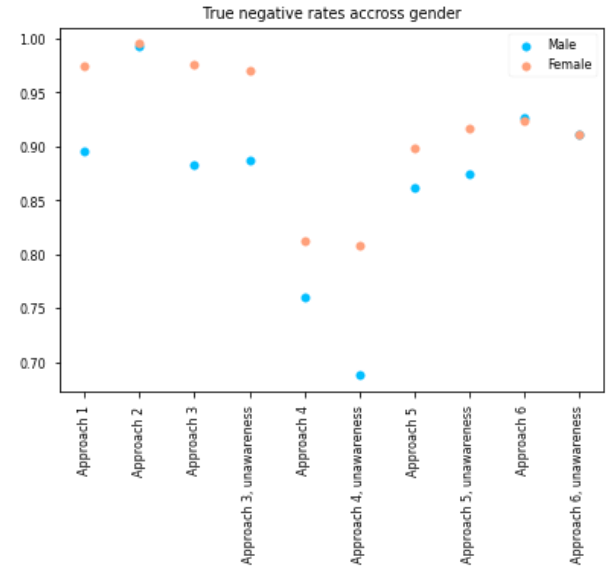

Figure 46: Comparing true negative rates across gender

5.1.5 Comparing True positive rate on positive predictions and true negative rate on negative predictions

The plots shown in Figures 48 and 49 show the comparison between the true positive rate on positive predictions across gender and the true negative rate on negative predictions across gender attributes for all the techniques.

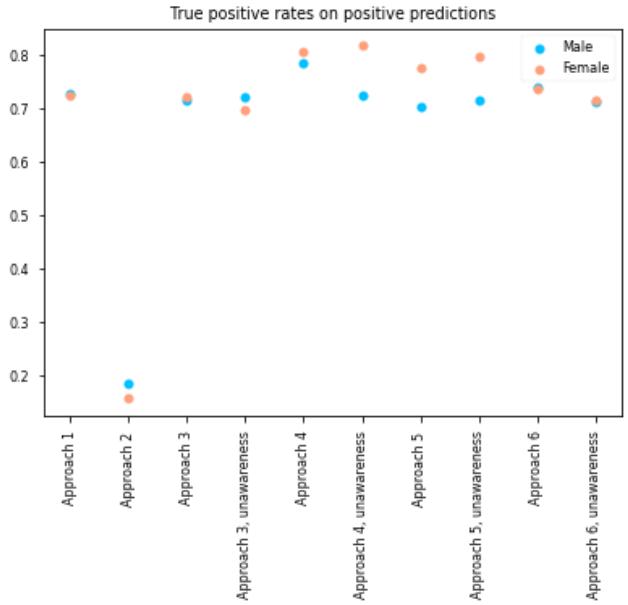

Figure 49: Comparing true positive rates on positive predictions

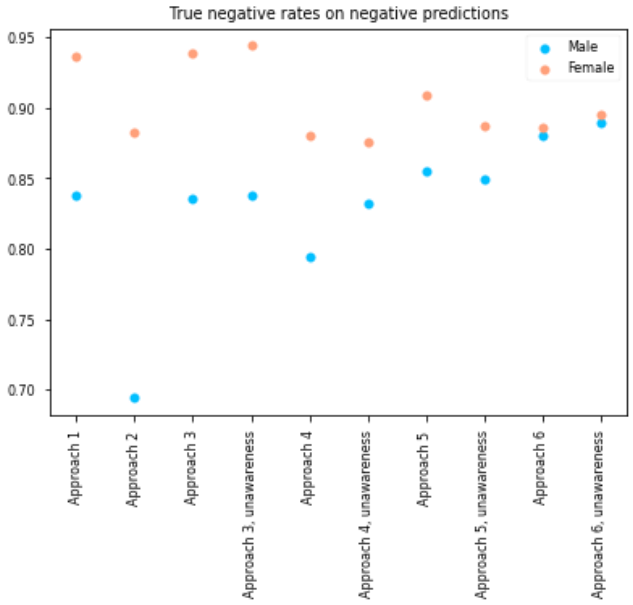

Figure 48: Comparing true negative rates on negative predictions

5.2 Receiver Operating Curve (ROC)

5.2.1 Receiver Operating Curve (ROC) across the model

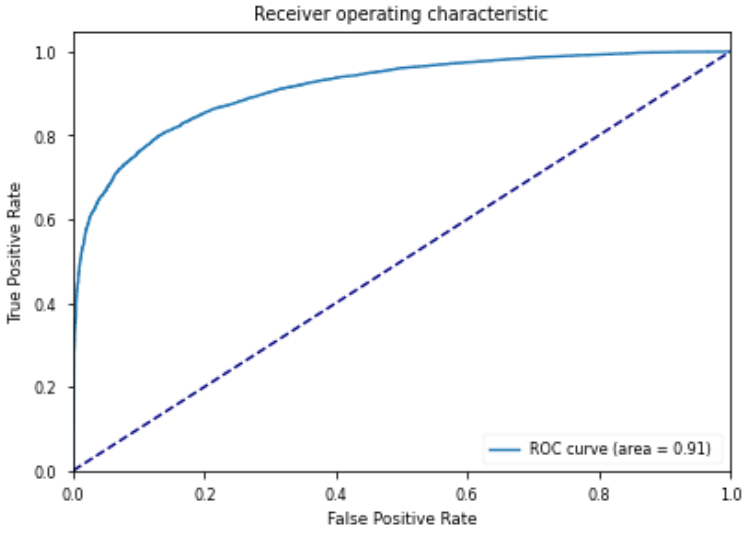

Figure 50: Receiver Operating Characteristic (ROC) Curve across model

5.2.2 Receiver Operative Curve (ROC) across gender

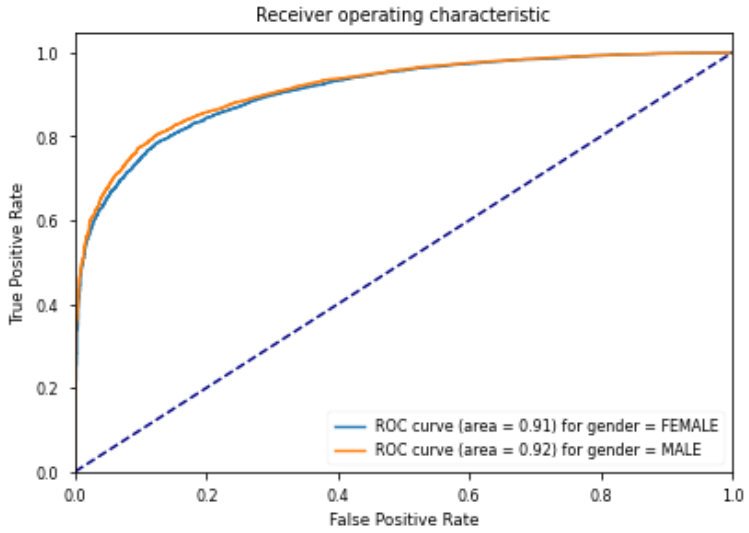

Figure 51: Receiver Operating Characteristic (ROC) Curve across gender

5.3 Single trained models

5.3.1 Comparing models for overall accuracy

The plot shown in Figure 52 shows the result for overall accuracy. We can observe that the random forest classifier has the highest accuracy, about 96% or 0.96. We can also observe that the Gaussian Naïve Bayes model has an accuracy of about 72% or 0.72. It can be observed that all the other models fall in between.

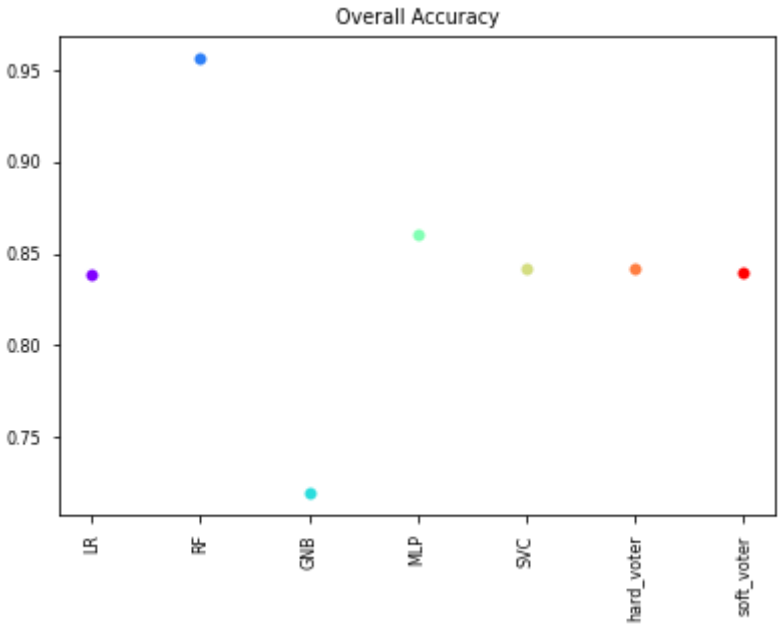

Figure 52: Comparing single trained models for overall accuracy

5.3.2 Comparing models for overall accuracy across gender

The plot shown in Figure 53 shows overall accuracy across gender. There are varying levels of the gap between the male and female demographic, depending on the model type. This shows that there is a bias in different models.

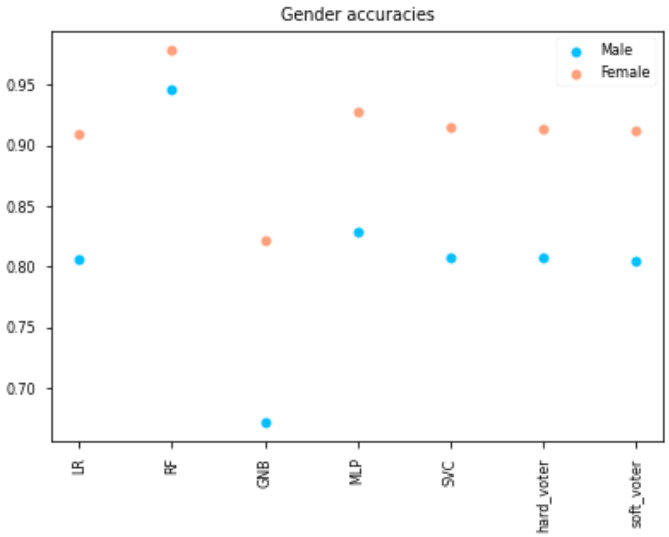

Figure 53: Comparing single trained models for overall accuracy across gender

5.3.3 Comparing positive and negative rates across gender

The following plots in Figures 54 and 55 show the positive and the negative rates across gender. Figure 55 shows that the positive rate for the male demographic is always higher than the female demographic. This concept can be problematic if we try to implement these models in the real world. In such a scenario, the model might predict a better outcome for the male demographic.

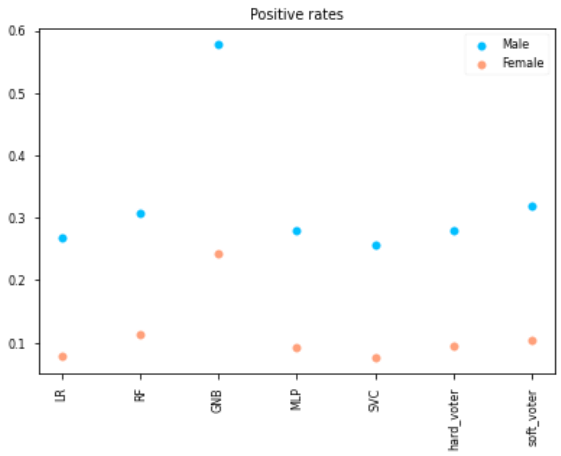

Figure 55: Comparing single trained models for positive rates across gender

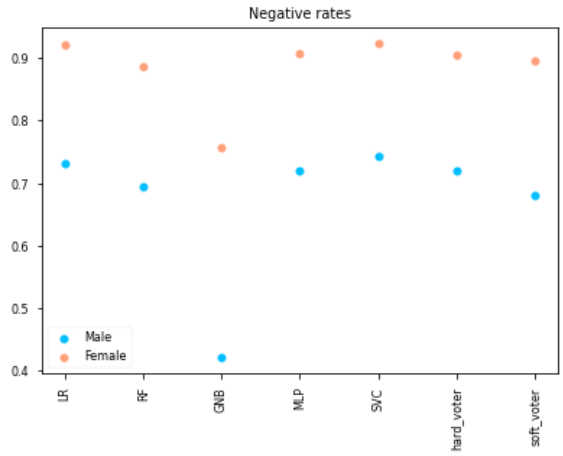

Figure 54: Comparing single trained models for negative rates across gender

5.3.4 Comparing true positive rate and true negative rate across gender

The following plots in Figures 56 and 57 show the true positive and true negative rates across gender. In Figure 57, we can see that the true positive rate is higher for males than for females. Furthermore, the true negative rate is higher for women than for men, as Figure 56 shows that the female demographic has a higher true negative rate than the male demographic. This can especially be problematic since it shows that our models cannot classify better-off high-income female earners as high-income male earners. Furthermore, classifying low-income female earners over low-income male earners could widen the gap between genders.

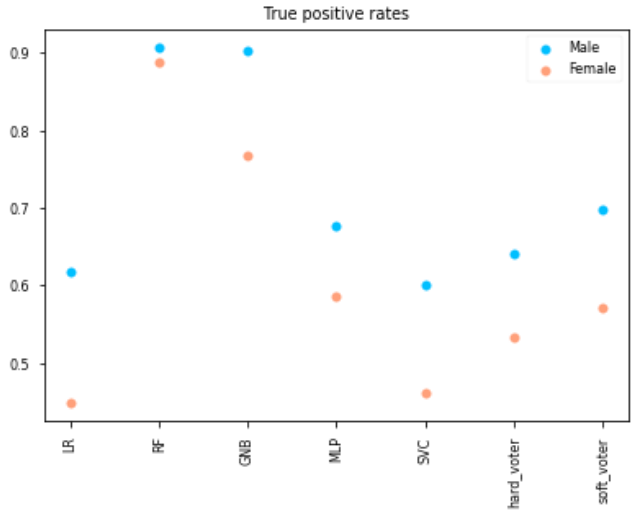

Figure 57: Comparing single trained models for true positive rates across gender

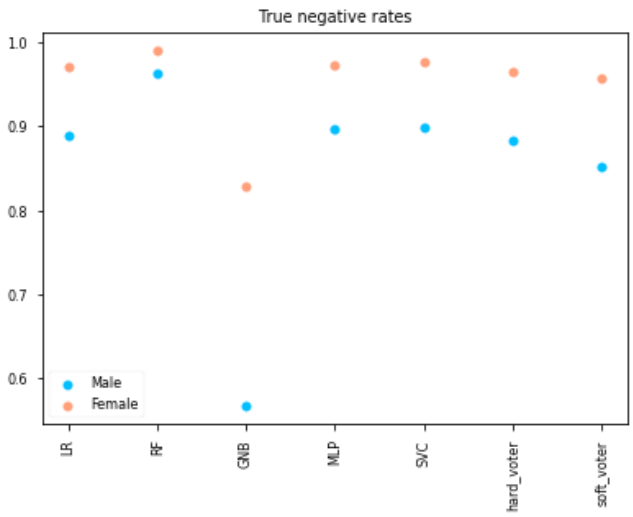

Figure 56: Comparing single trained models for true negative rates across gender

5.4 Multi-trained models

For the experiment, we ran it five times. We then compared the results across multiple training sessions to better understand the model's behaviour. We evaluated the relative value of the differences in the various metrics of interest between the female and male demographics for each of these cases.

5.4.1 Comparing difference of accuracy across models

Looking at the plot in Figure 58 for the accuracy disparity comparison, it can be observed that the models like Logistic Regression or hard voting or Support Vector Classifier (SVC) have a significantly lower accuracy disparity than Gaussian Naïve Bayes (GNB) or Random Forest.

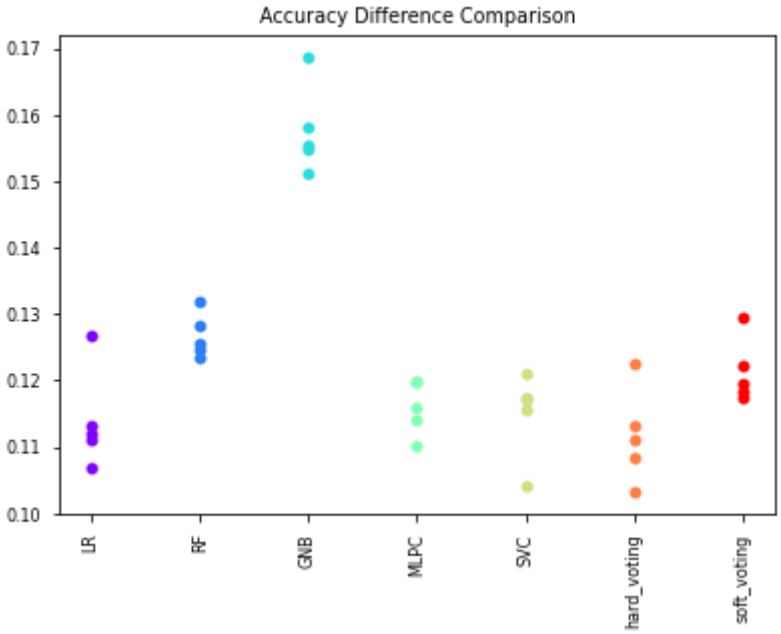

Figure 58: Comparing multi-trained models for accuracy

5.4.3 Comparing difference of positive and negative rates across models

It can be observed in Figures 59 and 60 by looking at the positive and the negative rate disparity. Suppose we look at models like Logistic Regression, Support Vector Classifier (SVC), or hard voting. In that case, it can be observed that they have significantly lower disparity than Gaussian Naïve Bayes (GNB).

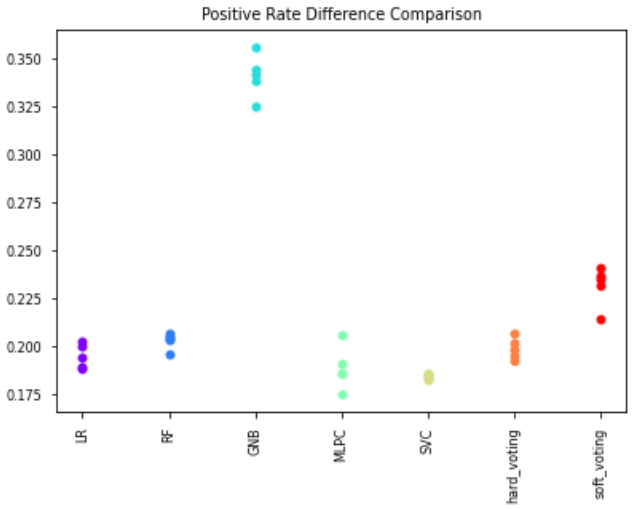

Figure 60: Comparing multi-trained models for positive rate difference

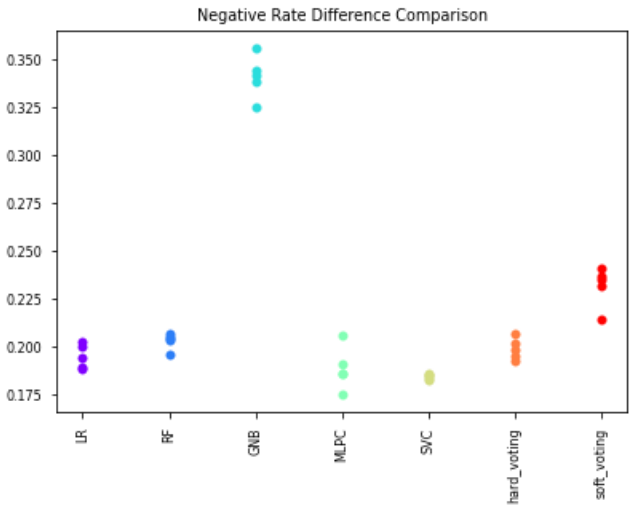

Figure 59: Comparing multi-trained models for negative rate difference

5.4.4 Comparing difference of true positive rate and true negative rate across models

We observe a substantially different result in Figures 61 and 62. Let us look at the true positive rate disparity in Figure 62. it can be observed that the models like logistic

regression or SVC now have higher disparity and higher variability than the models like GNB.

However, looking at the plot for the true negative rate in Figure 61, it can be observed that the plot tends to follow the preceding trend, wherein logistic regression and SVC have decreased variability and decreased disparity than GNB.

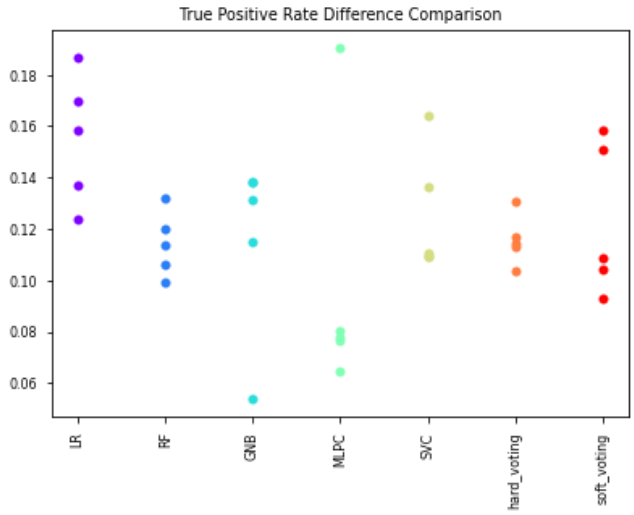

Figure 62: Comparing multi-trained models for true positive rate difference

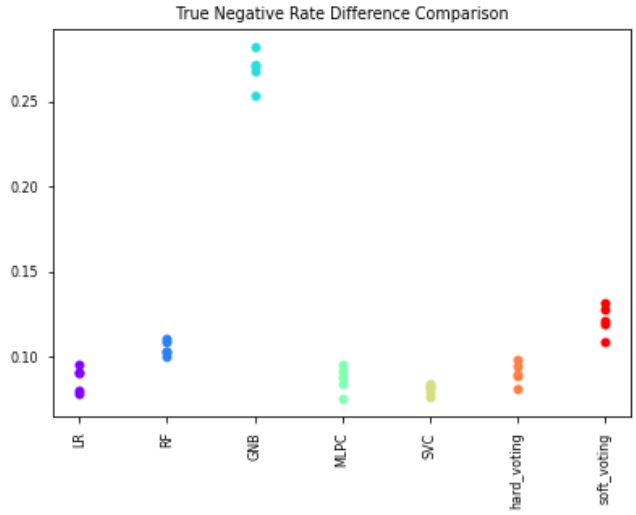

Figure 61: Comparing multi trained models for true negative rate difference

6.0 Conclusion

The algorithmic fairness line of research in machine learning has been around for a long time. The literature survey revealed that digital technologies that can be prone to algorithmic bias are increasing in various sectors. In this report, we considered exploring the bias present in the UCI Adult data set. We have tried various intervention strategies to improve fairness in the UCI Adult dataset by identifying various metrics commonly used in machine learning to measure fairness. Post-processing techniques are the easiest to implement as they do not impose restrictions on the model or the complexity of the work. For demographic parity, the support vector classification method and the equalised odds post-processing intervention were very effective.

6.1 Limitations and future works

The literature on algorithmic bias is full of speculative claims and is rarely based on solid evidence. This is mainly due to the lack of systematic review tools that allow researchers to examine the algorithms used by algorithmic scientists (Garcia-Gathright. Et al., 2018). Many companies are very protective of their algorithms and are willing to risk their reputations if competitors replicate them. For instance, the exact composition of Google's PageRank algorithm is a closely guarded secret (DeMers, 2018). The debate about algorithmic bias is often intense in some areas. This report tackles the topic in more detail, focusing on how to mitigate it in various sectors. Although some of the issues may appear in different sectors, they all have specific impacts on society. For instance, the effects of bias in crime and justice could be higher for a person than for a job application. The distinction between the most desirable outcome and the fairest outcome is not always obvious. However, it is often not immediately apparent what would constitute the most advantageous outcome. The increasing multi-disciplinary interest in the topic has raised various concerns. Among these is the need for a coherent framework for future research.

7.0 References

Abdollahi, B. and Nasraoui, O., 2018. Transparency in Fair Machine Learning: The Case of Explainable Recommender Systems. Human and Machine Learning, pp.21-35.

Alpaydin, E., 2014. Introduction to machine learning. Cambridge: MIT Press.

Baeza-Yates, R., 2018. Bias on the web. Communications of the ACM, 61(6), pp.54-61.

Baeza-Yates, R., 2020. Bias in search and recommender systems. In: Fourteenth ACM conference on recommender systems. New York: Association for Computing Machinery, p.2.

Bishop, C., 2016. Pattern recognition and machine learning. New York: Springer.

Biswas, S. and Rajan, H., 2020. Do the machine learning models on a crowd sourced platform exhibit bias? an empirical study on model fairness. Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering.

Burke, R., Sonboli, N. and Orgonez-Gauger, A., 2018. Balanced neighborhoods for multi-sided fairness in recommendation. FAT (2018),.

Cerda, P., Varoquaux, G. and Kégl, B., 2018. Similarity encoding for learning with dirty categorical variables. Machine Learning, 107(8-10), pp.1477-1494.

Corbett-Davies, S., Pierson, E., Feller, A., Goel, S. and Huq, A., 2017. Algorithmic decision making and the cost of fairness. *arXiv*, [online] (4). Available at: <https://arxiv.org/abs/1701.08230v4> [Accessed 3 September 2021].

Crabbé, A. H., Cahy, T., Somers, B., Verbeke, L.P., Van Coillie, F., 2020. Neural Network MLPClassifier (Version x.x) [Software]. Available from https://bitbucket.org/kul-reseco/lnns.

DeMers, J., 2018. *How Much Do We Really Know About Google's Ranking Algorithm?*. [online] Forbes. Available at: <https://www.forbes.com/sites/jaysondemers/2018/02/07/how-much-do-we-really-know-about-googles-ranking-algorithm/#30fa9fc955bb> [Accessed 3 September 2021].

Feldman, M., Friedler, S., Moeller, J., Scheidegger, C. and Venkatasubramanian, S., 2015. Certifying and Removing Disparate Impact. Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.

Fish, B., Kun, J. and Lelkes, Á., 2016. A Confidence-Based Approach for Balancing Fairness and Accuracy. Proceedings of the 2016 SIAM International Conference on Data Mining,.

Freedman, D., Pisani, R. and Purves, R., 2007. Statistics. New York: W.W. Norton & Co.

Friedman, B. and Nissenbaum, H., 1996. Bias in computer systems. ACM Transactions on Information Systems, 14(3), pp.330-347.

Garcia-Gathright, J., Springer, A., & Cramer, H., 2018. Assessing and Addressing Algorithmic Bias - But Before We Get There. *ArXiv*, *abs/1809.03332*.

Hardesty, L., 2018. *Study finds gender and skin-type bias in commercial artificial-intelligence systems*. [online] MIT News | Massachusetts Institute of Technology. Available at: <http://news.mit.edu/2018/study-finds-gender-skin-type-bias-artificial-intelligence-systems-0212> [Accessed 3 September 2021].

Hardt, M., Price, E. and Srebro, N., 2016. Equality of Opportunity in Supervised Learning. [online] 1, p.4. Available at: <https://arxiv.org/pdf/1610.02413.pdf> [Accessed 30 August 2021].

Hastie, T., Tibshirani, R. and Friedman, J., 2008. The elements of statistical learning. 2nd ed. New York: Springer, p.134.

He, H. and Ma, Y., 2013. Imbalanced learning. 1st ed. p.29.

“Impact of Autonomous Vehicle Technology on Long Distance Travel Behavior.” IIE Annual Conference. Proceedings, Institute of Industrial and Systems Engineers (IISE), Jan. 2020, p. 115.

Kamiran, F., Karim, A. and Zhang, X., 2012. Decision Theory for Discrimination-Aware Classification. 2012 IEEE 12th International Conference on Data Mining,.

Kamishima, T., Akaho, S., Asoh, H. and Sakuma, J., 2012. Fairness-Aware Classifier with Prejudice Remover Regularizer. Machine Learning and Knowledge Discovery in Databases, pp.35-50.

Liang, J., 2011. Predicting borrowers’ chance of defaulting on credit loans. Postgraduate. Stanford University.

Pedreshi, D., Ruggieri, S. and Turini, F., 2008. Discrimination-aware data mining. Proceeding of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD 08, pp.4-5; pp.560-568.

Piryonesi, S. and El-Diraby, T., 2020. Role of Data Analytics in Infrastructure Asset Management: Overcoming Data Size and Quality Problems. Journal of Transportation Engineering, Part B: Pavements, 146(2), p.04020022.

Prater, C., 2010. Applying for a credit card after Feb. 22? Issuers will be checking income - CreditCards.com. [online] CreditCards.com. Available at: <https://www.creditcards.com/credit-card-news/credit-card-application-income-check-1282/> [Accessed 19 August 2021].

Robinson, S., 2020. Understanding how deep learning black box training creates bias. [online] SearchEnterpriseAI. Available at: <https://searchenterpriseai.techtarget.com/feature/Understanding-how-deep-learning-black-box-training-creates-bias> [Accessed 1 September 2021].

Sweeney, L., 2013. *Discrimination in Online Ad Delivery*. [online] SSRN. Available at: <https://dx.doi.org/10.2139/ssrn.2208240> [Accessed 3 September 2021].

Truss, L., 2020. Fight For Fairness. [online] GOV.UK. Available at: <https://www.gov.uk/government/speeches/fight-for-fairness> [Accessed 21 August 2021].

Verma, S. and Rubin, J., 2018. Fairness definitions explained. Proceedings of the International Workshop on Software Fairness,.

Vincent, J., 2018. *Amazon reportedly scraps internal AI recruiting tool that was biased against women*. [online] The Verge. Available at: <https://www.theverge.com/2018/10/10/17958784/ai-recruiting-tool-bias-amazon-report> [Accessed 3 September 2021].

Walker, S. and Duncan, D., 1967. Estimation of the Probability of an Event as a Function of Several Independent Variables. Biometrika, 54(1/2), p.167.

Wasserman, P. and Schwartz, T., 1988. Neural networks. II. What are they and why is everybody so interested in them now?. IEEE Expert, 3(1), pp.10-15.

Xiao, Y., Wu, J., Lin, Z. and Zhao, X., 2018. A deep learning-based multi-model ensemble method for cancer prediction. Computer Methods and Programs in Biomedicine, 153, pp.1-9.

Zemel, R., Wu, Y., Swersky, K., Pitassi, T., & Dwork, C.,2020. Learning fair representations. In Proceedings of the 30th international conference on machine learning, pp. 325–333.

Zhou, J. & Chen, F. 2018, "Human and machine learning e-book", , pp. 21-35.

8.0 Appendix

8.1 Code for understanding the UCI Adult data set

```
# Load needed modules
%matplotlib inline
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import warnings
warnings.filterwarnings('ignore')


# Load the data set
adult_train_path = "https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.data"
adult_test_path = "https://archive.ics.uci.edu/ml/machine-learning-
databases/adult/adult.test"

col_names = ["age", "workclass", "fnlwgt", "education", "education-num",
"marital-status",
             "occupation", "relationship", "race", "sex", "capital-gain",
"capital-loss",
             "hours-per-week", "native-country", "salary"]

continious_features = ['age', 'fnlwgt', 'education-num', 'capital-gain',
'capital-loss',
                       'hours-per-week']

categorical_features = ["workclass", "education", "marital-status",
"occupation",
                        "relationship", "race", "sex", "native-country",
"salary"]

adult_train_data = pd.read_csv(adult_train_path, header=None,
names=col_names)
adult_test_data = pd.read_csv(adult_test_path, header=None,
names=col_names, skiprows=[0])
data = pd.concat([adult_train_data, adult_test_data])
data.reset_index(inplace = True, drop = True)

data['salary'].replace(regex=True,inplace=True,to_replace=r'\.',value=r'')
data.tail(5)


# Get the shape of the data
data.shape


# Function for plotting categorical variables
def plot_categs(df, category, fignum=1, title="Histogram of number of
datapoints"):
    plt.figure(fignum)
    uniques = list(sorted(df[category].unique()))
    counts = [df[df[category] == value].shape[0] for value in uniques]
```

```
    size = len(uniques)
    xcoords = list(range(1, size+1))
    plt.style.use('seaborn-paper')
    plt.bar(xcoords, counts)
    plt.xticks(xcoords, uniques, rotation='vertical' if size >= 5 else
'horizontal')
    plt.title((title if title is not None else ''))
    plt.tight_layout()


# Plot sex, race and native country

features = ['sex', 'race', 'native-country']
for index, feature in enumerate(features):
    plot_categs(data, feature, fignum=index+1)


# Plot education level and occupation

other_features = ['education', 'occupation']
for index, feature in enumerate(other_features):
    plot_categs(data, feature, fignum=index+1)

# Plot salary

plot_categs(data, 'salary', title="Histogram of number of datatopoints per
income level")


# Function to explore salary across gender

def subplot_categs(dfs, titles, category, fignum=1):
    plt.figure(fignum, figsize=(12, 6))
    number_of_dfs = len(titles)
    first_axis = None
    for df_index, df in enumerate(dfs):
        title = titles[df_index]
        uniques = list(sorted(df[category].unique()))
        counts = [df[df[category]==value].shape[0] for value in uniques]
        size = len(uniques)
        xcoords = list(range(1, size+1))
        if df_index == 0:
            first_axis =plt.subplot(1, 2, df_index+1)
        else:
            new_axis = plt.subplot(1, 2, df_index + 1, sharey=first_axis)
        plt.bar(xcoords, counts)
        plt.xticks(xcoords, uniques, rotation='vertical' if size >= 5 else
'horizontal')
        plt.title((title if title else ''))
        plt.tight_layout()


# Examining income category distribution among the male and female
demographics
male_data = data[data.sex == ' Male']
female_data = data[data.sex == ' Female']
titles = ['Histogram of male population per income category.', 'Histogram
of female population per income category.']
subplot_categs([male_data, female_data], titles, 'salary')
```

```
# Examine races in the dataset
plot_categs(data, 'race')

unique_races = data['race'].unique()
for index, race in enumerate(unique_races):
    plot_categs(data[data['race'] == race], 'salary', fignum=index,
                title = "Distribution of salary among the population
of"+race+" race")


# Function to plot hours-per-week

def plot_categ_hists(df, feature, title=None, bins=None, histtype='bar',
density=False, normed=None, fignum=1):
    plt.figure(fignum)
    x = df[feature]
    plt.hist(x, bins=bins, density=density, histtype=histtype)
    plt.xlabel(feature)
    plt.title(title if title is not None else '')
plot_categ_hists(data, 'hours-per-week', normed=True, fignum=1)
```

8.2 Code for preparing data for predictions

```
# Create a new dataset to be modified
datav2 = data.copy()


# See features with question marks
[feature for feature in datav2.columns if ' ?' in datav2[feature].unique()
or np.nan in datav2[feature].unique()]

# Remove all missing values
for feature in datav2.columns:
    datav2[feature] = datav2[feature].replace(' ?', np.nan)
datav2.dropna(how='any', inplace=True)

datav2['native-country'].unique(), datav2['workclass'].unique(),
datav2['occupation'].unique()


# Converting native country to binary one-hot for US vs non-US
datav2[datav2['native-country'] == ' United-States'].shape
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.tail(5)


# Converting gender and salary to binary one-hot

FEMALE_LABEL, MALE_LABEL = (0, 1)
HIGH_SALARY_LABEL, LOW_SALARY_LABEL = (0, 1)
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.tail(5)


# Transforming marital status to single or couple

datav2['marital-status'].unique()
datav2['marital-status'] = datav2['marital-status'].replace([' Divorced','
Married-spouse-absent',' Never-married',' Separated',' Widowed'],'Single')
datav2['marital-status'] = datav2['marital-status'].replace([' Married-AF-
spouse',' Married-civ-spouse'],'Couple')
datav2.head(-5)


# Convert relationships to binary one-hot

COUPLE_STATUS_LABEL, SINGLE_STATUS_LABEL = (0, 1)
datav2['marital-status'] = datav2['marital-
status'].map({'Couple':COUPLE_STATUS_LABEL,'Single':SINGLE_STATUS_LABEL})
datav2.tail(5)
```

```
# Converting relationship to one-hot

# 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.tail(10)
# Now convert relationship from integer to one-hot
datav2 = pd.get_dummies(datav2, columns=['relationship'])
datav2.tail(5)


# Converting race to one-hot

# Convert to integers first
race_map={' White':0,' Amer-Indian-Eskimo':1,' Asian-Pac-Islander':2,'
Black':3,' Other':4}
datav2['race']= datav2['race'].map(race_map)
# Convert from integer to one_hot
datav2 = pd.get_dummies(datav2, columns=['race'])
datav2.tail(5)


# Transform workclass feature

def group_workclass(x):
    if x['workclass'] == ' Federal-gov' or x['workclass']== ' Local-gov' or
x['workclass']==' State-gov': return 'govt'
    elif x['workclass'] == ' Private':return 'private'
    elif x['workclass'] == ' Self-emp-inc' or x['workclass'] == ' Self-emp-
not-inc': return 'self_employed'
    else: return 'without_pay'

datav2['workclass']=datav2.apply(group_workclass, axis=1)

datav2['workclass'] =
datav2['workclass'].map({'govt':0,'private':1,'self_employed':2,'without_pa
y':3})
datav2 = pd.get_dummies(datav2, columns=['workclass'])
datav2.tail(5)


# Converting occupation to one-hot

occupation_map = dict((value, key) for (key, value) in
enumerate(datav2.occupation.unique()))
datav2['occupation'] = datav2['occupation'].map(occupation_map)
datav2 = pd.get_dummies(datav2, columns=['occupation'])
datav2.tail(5)


# Converting capital-gain and capital-loss to categorical

capital_gain, capital_loss, =datav2['capital-gain'], datav2['capital-loss']
plt.hist(capital_gain,bins=None)
plt.show()
datav2.loc[(datav2['capital-gain'] > 0),'capital-gain'] = 1
datav2.loc[(datav2['capital-gain'] == 0 ,'capital-gain')]= 0
datav2.tail(5)
```

```
x = datav2['capital-loss']
plt.hist(x,bins=None)
plt.show()

datav2.loc[(datav2['capital-loss'] > 0),'capital-loss'] = 1
datav2.loc[(datav2['capital-loss'] == 0 ,'capital-loss')]= 0
datav2.tail(5)
```

8.3 Code for predicting individual's income without auditing bias

```
# Normalize continuous features
continuous_features = ['age', 'fnlwgt', 'education-num', 'hours-per-week']
X = datav2[continuous_features]
datav2[continuous_features] = (X - np.mean(X))/ np.std(X)
datav2.tail(5)


# Create a copy of the data set

datav2 = datav2.drop(['education'], axis=1)


# Training a machine learning algorithm on the data

from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
def get_naive_dataset(dataset):
    data_shuffled = dataset.sample(frac=1).reset_index(drop=True)
    X = data_shuffled.drop(['salary'], axis=1)
    y = data_shuffled['salary']
    x_train, x_test, y_train, y_test = train_test_split(X, y,
test_size=0.25)
    return (x_train, y_train), (x_test, y_test)
MLP_MAX_ITER=1000
(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)


# Evaluating algorithms performance

test_df = x_test.copy()
test_df['salary'] = y_test
test_df['pred'] = pd.Series(prediction, index=test_df.index)
test_df['accurate'] = (test_df['pred'] == test_df['salary'])
test_df.tail(5)

"Accuracy: ", test_df.accurate.mean()

# Function to understand gender bias in machine learning predictions
def evaluate_gender_performance(results_df, print_stats=False):
    """
    Evaluating gender performance
        - DRYing [Dont Repeat Yourself]

    Returns
        - Overall accuracy
        - Accuracy across gender
        - Positive rates
        - Negative rates
        - True positive and True negative rates
    """
    def printline():
        _print('-----------------------------------------------------')

    def _print(*args, **kwargs):
```

```
        if print_stats:
            print (args, kwargs)

    summaries = {}
    overall_accuracy = results_df.accurate.mean()
    summaries['accuracy_overall'] = overall_accuracy
    printline()
    _print("\n1.Overall accuracy: ", overall_accuracy)

    printline()

    # Accuracy accross gender
    _print("\n2.Accuracy accross gender \n ")
    printline
    for gender in [(FEMALE_LABEL, "Female"), (MALE_LABEL, "Male")]:
        rows = results_df[results_df.sex==gender[0]]
        accuracy_for_gender = rows.accurate.mean();
        summaries['accuracy_'+gender[1]] = accuracy_for_gender
        _print("P(((high, HIGH) or (low, LOW)) |", gender[1], "): ",
accuracy_for_gender)
    printline()

    _print("\n3.Positive Rates: \n")
    # High income rate given gender
    for gender in [(FEMALE_LABEL, "Female"), (MALE_LABEL, "Male")]:
        rows = results_df[results_df.sex==gender[0]]
        positive_rate_for_gender = (rows['pred']
==HIGH_SALARY_LABEL).mean()
        summaries['positive_rate_'+gender[1]] = positive_rate_for_gender
        _print("P(high|", gender[1], "): ", positive_rate_for_gender)
    printline()

    _print("\n4. Negative Rates: \n")

    # Low income rate given gender
    # High income rate given gender
    for gender in [(FEMALE_LABEL, "Female"), (MALE_LABEL, "Male")]:
        rows = results_df[results_df.sex==gender[0]]
        positive_rate_for_gender = (rows['pred'] ==LOW_SALARY_LABEL).mean()
        summaries['negative_rate_'+gender[1]] = positive_rate_for_gender
        _print("P(low|", gender[1], "): ", positive_rate_for_gender)
    printline()

    _print("\n4. True positive and True negative rates")

    printline()

    for index, gender in enumerate([(FEMALE_LABEL, "Female"), (MALE_LABEL,
"Male")]):

        _print("\n4."+("i")*(1+index), " True positive and negative rates
on sex="+gender[1], "\n")
        rows = results_df[results_df.sex==gender[0]]

        high_income = rows[rows.salary== HIGH_SALARY_LABEL]
        low_income=rows[rows.salary == LOW_SALARY_LABEL]
        if high_income.shape[0] > 0:
```

```
            assert high_income.salary.mean() == HIGH_SALARY_LABEL,
"high_mean: " + str(high_income.salary.mean())
        if low_income.shape[0] > 0:
            assert low_income.salary.mean() == LOW_SALARY_LABEL, "low_mean:
" + str(low_income.salary.mean())

        high_pred = rows[rows.pred == HIGH_SALARY_LABEL]
        low_pred = rows[rows.pred == LOW_SALARY_LABEL]
        if high_pred.shape[0] > 0:
            assert high_pred.pred.mean() == HIGH_SALARY_LABEL,
"high_pred_mean: " + str(high_pred.pred.mean())
        if low_pred.shape[0] > 0:
            assert low_pred.pred.mean() == LOW_SALARY_LABEL,
"low_pred_mean: " + str(low_pred.pred.mean())

        printline()
        true_positive_rate = high_income.accurate.mean()
        true_negative_rate = low_income.accurate.mean()
        summaries['true_positive_rate_'+gender[1]] = true_positive_rate
        summaries['true_negative_rate_'+gender[1]] = true_negative_rate

        _print(str.format("P((high, HIGH)| HIGH,{0})", gender[1]), ":
",true_positive_rate)
        _print(str.format("P((low, LOW)| LOW,{0})", gender[1]),
":",true_negative_rate)

        printline()
        true_positive_rate_on_positive_predictions =
high_pred.accurate.mean()
        true_negative_rate_on_negative_predictions =
low_pred.accurate.mean()
        summaries['true_positive_rate_on_positive_predictions_'+gender[1]]
= true_positive_rate_on_positive_predictions
        summaries['true_negative_rate_on_negative_predictions_'+gender[1]]
= true_negative_rate_on_negative_predictions
        _print(str.format("P((high,HIGH) | high, {0})", gender[1]), ":
",true_positive_rate_on_positive_predictions)
        _print(str.format("P((low,LOW) |low,{0})", gender[1]), ":",
true_negative_rate_on_negative_predictions)

    return summaries

def plot_performance_per_group(accuracy_results, title, fignum=1,
rotation='horizontal', labels=["Male", "Female"]):

    """
    Plot results for 2 groups stacked together
    """
    assert isinstance(accuracy_results, list), "Accuracy results must be a
list"


    indices = [0]
    colors = ['deepskyblue', 'lightsalmon']
    fig, ax = plt.subplots()

    for index in indices:
```

```
        ax.scatter(index, accuracy_results[0][index], c=colors[0],
label=labels[0] if labels and index ==0 else None)
        ax.scatter(index, accuracy_results[1][index], c=colors[1],
label=labels[1] if labels and index ==0 else None)

    if labels:
        ax.legend()

    #plt.xticks(indices, approaches, rotation=rotation)
    plt.title(title)

    plt.show()


def plot_comparisons_groups(approaches, accuracy_results, title, fignum=1,
rotation='horizontal', labels=["Male", "Female"]):
    """
    Plot results for 2 groups stacked together
    """
    assert isinstance(accuracy_results, list), "Accuracy results must be a
list"


    indices = list(range(len(approaches)))
    colors = ['deepskyblue', 'lightsalmon']
    fig, ax = plt.subplots()

    for index in indices:
        ax.scatter(index, accuracy_results[0][index], c=colors[0],
label=labels[0] if labels and index ==0 else None)
        ax.scatter(index, accuracy_results[1][index], c=colors[1],
label=labels[1] if labels and index ==0 else None)

    if labels:
        ax.legend()

    plt.xticks(indices, approaches, rotation=rotation)
    plt.title(title)

    plt.show()

def plot_model_gender_metrics(_feature, _summaries, _modelNames, _title,
rotation='vertical'):
    gender_metrics = [[summary[_feature+'_Male'] for summary in
_summaries],
                      [summary[_feature+'_Female'] for summary in
_summaries]
                     ]
    plot_comparisons_groups(_modelNames,gender_metrics, _title,
rotation=rotation)

def model_summary(model_name, title, summary):
    summaries = []
    model_names = []

    for key in ["accuracy", "positive_rate", "negative_rate",
"true_positive_rate", "true_negative_rate"]:
```

```
        new_summary = {"accuracy_Male": summary[key+"_Male"],
"accuracy_Female": summary[key+"_Female"]}
        summaries.append(new_summary)
        model_names.append(key)
    plot_model_gender_metrics("accuracy", summaries, model_names,
model_name)
    #plot_model_gender_metrics(key, [summary], [model_name],
"Model="+model_name+", Metric="+key, rotation="horizontal")

# Plotting accuracy
original_approach = evaluate_gender_performance(test_df)
model_summary("[Approach 1] Metrics with bias", "", original_approach)

# Function for plotting learning curve
default_training_sizes = [1000, 5000, 10000, 15000, 20000, 25000, 30000,
35000, 40000, 45000]

from sklearn.model_selection import learning_curve

def plot_learning_curve(estimator, lrxtrain, lrytrain, train_sizes,
title=None):
    _train_sizes = []
    for size in train_sizes:
        if size <= lrxtrain.shape[0]*.65:
            _train_sizes.append(size)
        else:
            break
    train_sizes, train_scores, validation_scores = learning_curve(
                                                estimator, lrxtrain,
lrytrain, train_sizes = _train_sizes, scoring = 'neg_log_loss')
    train_scores_mean = -train_scores.mean(axis = 1)
    validation_scores_mean = -validation_scores.mean(axis = 1)

    plt.plot(train_sizes, train_scores_mean, label = 'Training error')
    plt.plot(train_sizes, validation_scores_mean, label = 'Validation
error')

    plt.ylabel('NLL')
    plt.xlabel('Training set size')
    plt.title(title)
    plt.legend()

# Plot learning curve
plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, "Learning curve [Approach 1]")
```

8.4 Code for Bias mitigation techniques

8.4.1 Code for Mitigation through unawareness

```
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
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)


# Function for evaluating performance

def evaluate_predictor_performance(predictions, x_test, y_test):
    """
    Returns summary statistics for the predictor's performance

    Input:
        - predictions: model's predictions on x_test
        - x_test: test input
        - y_test: test labels

    Requires:
        - predictor must have been fitted on x_train and y_train from the
same dataset

    Check method evaluate_gender_performance for more on the produced
summary statistics
    """
    testdata = x_test.copy()
    testdata['salary'] = y_test
    testdata['pred'] = pd.Series(predictions, index=x_test.index)
    testdata['accurate'] = (testdata['pred'] == testdata['salary'])
    return evaluate_gender_performance(testdata)

predictions = predictor.predict(x_test)
approach_1 = evaluate_predictor_performance(predictions, testdata, y_test)
model_summary("[Approach 2] Metrics of unawareness", "", approach_1)

plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curves [Approach 2]')
```

8.4.2 Code for Mitigation through data set balancing

8.4.2.1 Code for Equal number of data points per demographic

```
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)


# Creating a copy of the data set

datav3 = datav2.copy()
datav3.tail(5)

(x_train, y_train), (x_test, y_test) = get_gender_balanced_dataset(datav3)
x_train.shape, x_test.shape

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("[Approach 3] Mertics of equal samples of gender", "",
approach_2)

plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 3]')

predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
predictor.fit(x_train.drop(['sex'], axis=1), y_train)
approach_2_blind =
evaluate_predictor_performance(predictor.predict(x_test.drop(['sex'],
axis=1)), x_test, y_test)
model_summary("[Approach 3] Metrics of equal ample of gender and
unawareness", "", approach_2_blind)


plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve, [Approach 3, unawareness]')
```

8.4.2.2 Code for Equal number of data points per demographic in each category

```
# Function

def get_gender_category_balanced_dataset(dataset, test_size=0.25):
    """
    Equal number of datapoints per category. Limited by the smallest number
of points
    """
    # Old distribution categories
    males = dataset[(dataset.sex==MALE_LABEL)]
    females = dataset[(dataset.sex==FEMALE_LABEL)]
    male_high = males[(males.salary == HIGH_SALARY_LABEL)]
    male_low = males[(males.salary == LOW_SALARY_LABEL)]
    female_high = females[(females.salary == HIGH_SALARY_LABEL)]
    female_low = females[(females.salary == LOW_SALARY_LABEL)]

    # Smallest is the bottleneck
    smallest = min((x.shape[0] for x in [male_high, male_low, female_high,
female_low]))

    # New distribution categories
    _male_high = male_high.sample(n=smallest).reset_index(drop=True)
    _male_low = male_low.sample(n=smallest).reset_index(drop=True)
    _female_high = female_high.sample(n=smallest).reset_index(drop=True)
    _female_low = female_low.sample(n=smallest).reset_index(drop=True)
    _combined = pd.concat([_male_high, _male_low, _female_high,
_female_low]).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)


(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)


approach_3 = evaluate_predictor_performance(predictions, x_test, y_test)
model_summary("[Approach 4] Metrics of equal samples of income", "",
approach_3)


plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 4]')


(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.drop(['sex'], axis=1), y_train)
predictions = predictor.predict(x_test.drop(['sex'], axis=1))
```

```
approach_3_blind = evaluate_predictor_performance(predictions, x_test,
y_test)
model_summary("[Approach 4] Metrics of equal sample of income and
unawareness", "", approach_3_blind)


plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 4, unawareness]')
```

8.4.2.3 Code for Equal ratios instead of equal number of data points

```
# Function

def get_gender_category_ratio_balanced_dataset(dataset):

    """
    Ratio of (male_high, male_row) = Ratio of (female_high, female_low),
maximize number of real datapoints
    """

    # Old distribution categories
    males = dataset[(dataset.sex==MALE_LABEL)]
    females = dataset[(dataset.sex==FEMALE_LABEL)]
    assert males.shape[0] > 0 and females.shape[0] > 0, "Empty males or
females"

    male_high = males[(males.salary == HIGH_SALARY_LABEL)]
    male_low = males[(males.salary == LOW_SALARY_LABEL)]

    assert male_high.shape[0] > 0 and male_low.shape[0] > 0, " empty male
high or low"

    female_high = females[(females.salary == HIGH_SALARY_LABEL)]
    female_low = females[(females.salary == LOW_SALARY_LABEL)]

    assert female_high.shape[0] > 0 and female_low.shape[0] > 0, "empty
female high or low"


    print("shapes mh, ml, fh, fl: ", [x.shape[0] for x in [male_high,
male_low, female_high, female_low]])

    ratio = float(male_high.shape[0]) / float(male_low.shape[0])
    assert ratio > 0, " ratio must be greater than 0"

    print ("Ratio is ", ratio)
    n_female_high = female_high.shape[0]
    n_female_low = int(n_female_high / ratio)

    _male_low = male_low.copy()
    _male_high = male_high.copy()
    _female_high = female_high.copy()
    _female_low = female_low.sample(n=n_female_low).reset_index(drop=True)
    _combined = pd.concat([_male_high, _male_low, _female_high,
_female_low]).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=0.25)

    return (x_train, y_train), (x_test, y_test)

(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)

approach_4 = evaluate_predictor_performance(predictions, x_test, y_test)
model_summary("[Approach 5] Metrics of equal ratio of gender", "",
approach_4)

plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 5]')

predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
predictor.fit(x_train.drop(['sex'], axis=1), y_train)
predictions = predictor.predict(x_test.drop(['sex'], axis=1))
approach_4_blind = evaluate_predictor_performance(predictions, x_test,
y_test)
model_summary("[Approach 5] Metrics of equal ratio of gender with
unawareness", "", approach_4_blind)

plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 5, unawareness]')
```

8.4.3 Code for Mitigation through data augmentation

8.4.3.1 Code for Counterfactual augmentation

```
# Function

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

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("[Appraoch 6] Metrics of Counterfactual Augmentation", "",
ctf_1)


plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 6]')


predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
predictor.fit(x_train.drop(['sex'], axis=1), y_train)


ctf_blind =
evaluate_predictor_performance(predictor.predict(x_test.drop(['sex'],
axis=1)), x_test, y_test)
model_summary("[Approach 6] Counterfactual Augmentation with unawareness",
"", ctf_blind)


plot_learning_curve(MLPClassifier(), x_train, y_train,
default_training_sizes, 'Learning curve [Approach 6, unawareness]')
```

8.5 Code for comparing bias mitigation approach

```
# Function
def plot_comparisons(approach_names, accuracy_results, title, fignum=1,
rotation='horizontal'):
    """
    Args:
        - summary: Dictionary describing the approach's gender performance
        - approach_name: The name of the technique, to be displayed
    """
    assert isinstance(accuracy_results, list) and not
isinstance(accuracy_results[0], list), accuracy_results


    indices = list(range(len(approach_names)))
    colors = cm.rainbow(np.linspace(0, 1, len(indices)))
    plt.figure(fignum)

    for index in indices:
        plt.scatter(index, accuracy_results[index], color=colors[index])

    plt.xticks(indices, approach_names, rotation=rotation)

    plt.title(title)
    plt.show()

def plot_comparisons_groups(approaches, accuracy_results, title, fignum=1,
rotation='horizontal', labels=["Male", "Female"]):
    """
    Plot results for 2 groups stacked together
    """
    assert isinstance(accuracy_results, list), "Accuracy results must be a
list"


    indices = list(range(len(approaches)))
    colors = ['deepskyblue', 'lightsalmon']
    fig, ax = plt.subplots()

    for index in indices:
        ax.scatter(index, accuracy_results[0][index], c=colors[0],
label=labels[0] if labels and index ==0 else None)
        ax.scatter(index, accuracy_results[1][index], c=colors[1],
label=labels[1] if labels and index ==0 else None)

    if labels:
        ax.legend()

    plt.xticks(indices, approaches, rotation=rotation)
    plt.title(title)

    plt.show()

approaches = ['no_debias', 'gender_unaware', 'equal_|data|_per_gender',
'if_gender_blind', 'equal_data_per_(gender, category)', 'if_gender_blind'
'equal_data_ratio_per_gender', 'if_blind', 'ctf', 'ctf_blind']
```

```
summaries = [original_approach, approach_1, approach_2, approach_2_blind,
approach_3, approach_3_blind, approach_4, approach_4_blind, ctf_1,
ctf_blind]

# Comparing overall accuracies
accuracy_results = [summary['accuracy_overall'] for summary in summaries]
plot_comparisons(approaches, accuracy_results, 'Comparisons of overall
accuracy', rotation='vertical')

# Function for plotting
def plot_model_gender_metrics(_feature, _summaries, _modelNames, _title,
rotation='vertical'):
    gender_metrics = [[summary[_feature+'_Male'] for summary in
_summaries],
                      [summary[_feature+'_Female'] for summary in
_summaries]
                     ]
    plot_comparisons_groups(_modelNames,gender_metrics, _title,
rotation=rotation)


# Comparing overall accuracy across gender
plot_model_gender_metrics('accuracy', summaries, approaches, "Accuracy on
Male vs Accuracy on Female")

# Comparing positive and negative rates across gender
plot_model_gender_metrics('positive_rate', summaries, approaches, "Positive
rates accross gender")
plot_model_gender_metrics('negative_rate', summaries, approaches, "Negative
rates accross gender")

# Comparing true positive and negative rates across gender
plot_model_gender_metrics('true_positive_rate', summaries, approaches,
"True positive rates accross gender")
plot_model_gender_metrics('true_negative_rate', summaries, approaches,
"True negative rates accross gender")

# Comparing true positive rate on positive predictions, and true negative
rate on negative predictions
plot_model_gender_metrics('true_positive_rate_on_positive_predictions',
summaries, approaches, "True positive rates on positive predictions")
plot_model_gender_metrics('true_negative_rate_on_negative_predictions',
summaries, approaches, "True negative rates on negative predictions")

# Code for ROC and AUC curves
from sklearn import svm, metrics
clf = MLPClassifier(max_iter=MLP_MAX_ITER)
clf.fit(x_train, y_train)

# Function
def plot_roc_curve(trained_predictor, X_test_list=None, Y_test_list=None,
label_list = None, fignum=None):
    """
    Trained predictor must have .decision_function attribute
    """
    if fignum is not None:
        figure(fignum)
```

```
    for index in range(len(X_test_list)):
        X_test = X_test_list[index]
        Y_test = Y_test_list[index]
        assert X_test is not None and Y_test is not None, "X_test and
Y_test cannot be None"
        y_pred_scores = trained_predictor.predict_proba(X_test)[:, 1]
        fpr, tpr, thresholds = metrics.roc_curve(Y_test, y_pred_scores) #
obtain false positive and true positive rates
        area_under_curve = metrics.auc(fpr, tpr)
        label = "for gender = "+  label_list[index] if label_list is not
None else ''
        #plt.plot(fpr, tpr, color='darkorange', label='ROC curve (area =
%0.2f) %s' % (area_under_curve, label)) # plot ROC curve
        plt.plot(fpr, tpr, label='ROC curve (area = %0.2f) %s' %
(area_under_curve, label))
    plt.plot([0, 1], [0, 1], color='navy', linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('Receiver operating characteristic')
    plt.legend(loc="lower right")

plot_roc_curve(clf, X_test_list = [x_test], Y_test_list = [y_test])

predictor = MLPClassifier(max_iter=MLP_MAX_ITER)
predictor.fit(x_train, y_train)

# Function
def plot_gender_roc_curves(trained_predictor, X_test, Y_test):
    """
    Plots the ROC curve for each gender demographic
    """
    combined = pd.concat([X_test, Y_test], axis=1)
    x_test_list, y_test_list, gender_labels = [], [], []
    for gender, gender_label in (("FEMALE", FEMALE_LABEL), ("MALE",
MALE_LABEL)):
        with_gender = combined[combined['sex'] == gender_label]
        x_test = with_gender.drop(['salary'], axis=1)
        y_test = with_gender['salary']
        x_test_list.append(x_test)
        y_test_list.append(y_test)
        gender_labels.append(gender)
    plot_roc_curve(predictor, X_test_list=x_test_list,
Y_test_list=y_test_list,label_list=gender_labels)
plot_gender_roc_curves(predictor, x_test, y_test)
```

8.6 Code for bias mitigation through fair model selection

```
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

LR_MAX_ITER=1000

(x_train, y_train), (x_test, y_test) = get_naive_dataset(datav3)


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)


for model_name, model in [('LR', lr), ('RF', rf), ('GNB', gnb), ('MLP',
mlp), ('SVC', svc), ('KNC', knc)]:
    print(model_name, ' accuracy: ', accuracy_score(y_test,
model.predict(x_test)))
```

8.7 Code for debiasing through multi-model architecture

```
from sklearn.ensemble import VotingClassifier

# Function
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=voting)

    return voter

(x_train, y_train), (x_test, y_test) = get_naive_dataset(datav3)
```

In [ ]:

```
hardvoter = default_voting_classifier(voting='hard')
softvoter = default_voting_classifier(voting='soft')
for model in [hardvoter, softvoter]:
    model.fit(x_train, y_train)
print('Hard voting accuracy ', accuracy_score(y_test,
hardvoter.predict(x_test)))
print('Soft voting accuracy ', accuracy_score(y_test,
softvoter.predict(x_test)))

# Model persistence
import pickle

class Persistence:
    """
    Implements model persistence functionality
    """
    def __init__(self):
        pass

    @staticmethod
    def storeObject(_object, filename):
        pickle_out = open(filename,"wb")
        pickle.dump(_object, pickle_out)
        pickle_out.close()

    @staticmethod
    def loadObjects(filenames=None):
        result = []
        for filename in filenames:
            result.append(pickle.load(open(filename, 'rb')))
        return result

    @staticmethod
    def storeOrLoad(store=False, load=False, names=None, objects=None):
        """
        Returns file names if storing, returns objects if reading
        """
```

```
        if store or load:
            assert store != load, 'Cannot store and load'
        if store:
            for _object, name in zip(objects, names):
                Persistence.storeObject(_object, name)
            return 'Stored'
        if load:
            read = Persistence.loadObjects(filenames=names)
            return read

modelNames = ['LR', 'RF', 'GNB', 'MLP', 'SVC', 'hard_voter', 'soft_voter']
models = [lr, rf, gnb, mlp, svc, hardvoter, softvoter]


summaries = []
for model, modelname in zip(models, modelNames):
    summaries.append(evaluate_predictor_performance(model.predict(x_test),
x_test, y_test))
```

8.8 Code for comparing model performance for single training session

```
overall_accuracies = [summary['accuracy_overall'] for summary in summaries]
plot_comparisons(modelNames, overall_accuracies, "Overall Accuracy",
rotation='vertical')


gender_accuracies = [[summary['accuracy_Male'] for summary in summaries],
[summary['accuracy_Female'] for summary in summaries]]
plot_comparisons_groups(modelNames, gender_accuracies, "Gender accuracies",
rotation='vertical')


plot_model_gender_metrics('positive_rate', summaries, modelNames, "Positive
rates", rotation='vertical')
plot_model_gender_metrics('negative_rate', summaries, modelNames, "Negative
rates", rotation='vertical')


plot_model_gender_metrics('true_positive_rate', summaries, modelNames,
"True positive rates", rotation='vertical')
plot_model_gender_metrics('true_negative_rate', summaries, modelNames,
"True negative rates", rotation='vertical')
```

8.9 Code for comparing model performance over multiple training sessions

```
# Function

def get_model_class_summaries(model_class, dataset, training_sessions,
*args, **kwargs):
    """
    Repeatedly sample from the dataset, train, test and return summary
statistics
    """
    assert training_sessions >= 1, "Must train at least once"

    Xvals, Yvals = dataset
    summaries = []
    for session in range(training_sessions):
        x_train, x_test, y_train, y_test = train_test_split(Xvals, Yvals,
test_size=.25)
        model = model_class(*args, **kwargs)
        model.fit(x_train, y_train)

        evaluation = evaluate_predictor_performance(model.predict(x_test),
x_test, y_test)
        summaries.append(evaluation)

    assert len(summaries) == training_sessions
    return summaries


dataset = datav3.copy()
Xvals=dataset.drop(["salary"], axis=1)
Yvals = dataset["salary"]
some_summaries = get_model_class_summaries(MLPClassifier, (Xvals, Yvals),
2, max_iter=MLP_MAX_ITER)
some_summaries[:2]


single_model_name_classes_args_kwargs = [
    ['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,             [], {}],
    ['MLPC', MLPClassifier,          [], {'max_iter':MLP_MAX_ITER}],
    ['SVC', svm.SVC,                [], {}]
]


single_model_summaries = []


runLoop = True # simply set runLoop to False to avoid iterations
if runLoop:
    for name, model_class, args, kwargs in
single_model_name_classes_args_kwargs:
        model_class_summaries = get_model_class_summaries(model_class,
(Xvals, Yvals), 5, *args, **kwargs)
        single_model_summaries.append((name, model_class_summaries))
```

```
store = True # Change the flag to store the summaries
if store:
    Persistence.storeOrLoad(store=True, names=['single_model_summaries'],
objects = [single_model_summaries])


voting_model_name_classes_args_kwargs = [
    ['hard_voting', default_voting_classifier, [], {'voting':'hard'}],
    ['soft_voting', default_voting_classifier, [], {'voting':'soft'}]
]


voting_model_summaries = []


runLoop = True
if runLoop:
    for name, model_class, args, kwargs in
voting_model_name_classes_args_kwargs:
        model_class_summaries = get_model_class_summaries(model_class,
(Xvals, Yvals), 5, *args, **kwargs)
        voting_model_summaries.append((name, model_class_summaries))


assert [len(voting_model_summaries) > 0]
Persistence.storeOrLoad(store=True, names=['voting_model_summaries'],
objects=[voting_model_summaries])


def extract_treatment_differences(summaries):
    """
    Extract treatment difference(male-female) from a performance summary
    """
    differences = []
    for summary_dict in summaries:
        gender_attrs = set()
        for gender_key in summary_dict:
            if '_Male' in gender_key:
                gender_attrs.add(gender_key[:gender_key.rindex('_')])
        value_dict = {}
        for gender_attr in gender_attrs:
            value_dict[gender_attr] = summary_dict[gender_attr+"_Male"] -
summary_dict[gender_attr+ "_Female"]
        differences.append(value_dict)
    return differences


all_model_summaries = single_model_summaries + voting_model_summaries


store = True # set to True to store all model summaries
if store:
    Persistence.storeOrLoad(store=True, names=['all_model_summaries'],
objects=[all_model_summaries])


load = True # Set to true to load stored models
if load:
    all_model_summaries = Persistence.storeOrLoad(load=load,
names=['all_model_summaries'])[0]
```

```
all_model_summaries[-2:]


all_model_differences = []


for model_name, model_summaries in all_model_summaries:
    differences = extract_treatment_differences(model_summaries)
    all_model_differences.append((model_name, differences))


all_model_differences[:1]


def get_model_values_for_feature(feature, nsp, abs_val=False):
    """
    Inputs:
        nsp = name summary pairs
    """
    model_names = [model_summary[0] for model_summary in nsp]
    model_summary_lists = [model_summary[1] for model_summary in nsp] #
Each element is a list of dicts
    model_y_values = []
    for model_summary_list in model_summary_lists:
        values = [abs(model_summary[feature]) if abs_val else
model_summary[feature] for model_summary in model_summary_list]
        model_y_values.append(values)
    return model_names, model_y_values

def plot_model_values_for_feature(model_names, model_y_values, title,
rotation='vertical'):

    indices = list(range(len(model_names)))

    colors = cm.rainbow(np.linspace(0, 1, len(indices)))

    fig, ax = plt.subplots()

    for index in indices:
        for y_value in model_y_values[index]:
            ax.scatter(index, abs(y_value), color=colors[index],
label=model_names[index])
    plt.xticks(indices, model_names, rotation=rotation)
    #plt.yticks([0])
    plt.title(title)

    plt.show()

def plot_feature_differences(feature_name, model_differences, title):
    model_names, model_y_values =
get_model_values_for_feature(feature_name, all_model_differences)
    plot_model_values_for_feature(model_names, model_y_values, title)


plot_feature_differences('accuracy', all_model_differences, 'Accuracy
Difference Comparison')
```

```
pl = [('positive_rate', 'Positive Rate'), ('negative_rate', 'Negative
Rate')]
for feature, title in pl:
    plot_feature_differences(feature, all_model_differences, title+ '
Difference Comparison')


pl = [('true_positive_rate', 'True Positive Rate'), ('true_negative_rate',
'True Negative Rate')]
for feature, title in pl:
    plot_feature_differences(feature, all_model_differences, title+ '
Difference Comparison')
```