DA421: Fairness in AI Models

Assignment – 01 : Report

Name : Kaki Hephzi Sunanda

Roll Number: 210150018

Objective:

The objective of the assignment is to analyse bias in the adult (or census income) dataset with respect to the demographic attributes such as 'race' and 'marital-status'. This involves implementing the situation testing using k-nearest neighbor algorithm on the data to identify bias. Unfairness or discrepancies are detected and alleviated by using the DiscoveryN() and PreventionN() algorithms.

Two experiments are to be performed:

Experiment 1: Demographic attribute is 'race'

Experiment 2: Demographic attribute is 'marital-status'.

Additionally, C4.5 Decision Tree, Logistic Regression, Naïve Bayes classifiers are trained on the obtained pre-processed data. The accuracy and 0.10-discrimination metrics are calculated for each of the classifier and compared to the results of presented in the paper titled 'k-NN as an Implementation of Situation Testing for Discrimination Discovery and Prevention, SIGKDD, 2011'.

Dataset:

The dataset used in this assignment is the adult (also known as the census income) dataset. The following are the features in the dataset:

Variable name	Type	Demographic	Missing Values
age	Integer	Age	no
workclass	Categorical	Income	yes
fnlwgt	Integer		no
education	Categorical	Education Level	no
education-num	Integer	Education Level	no
marital-status	Categorical	Other	no
occupation	Categorical	Other	yes
relationship	Categorical	Other	no
race	Categorical	Race	no
sex	Binary	Sex	no
capital-gain	Integer		no
capital-loss	Integer		no
hours-per-week	Integer		no
native-country	Categorical	Other	yes
income	Binary	Income	no

The dataset is used to evaluate if there is a bias in the income based on the sensitive attributes 'race' and 'marital-status'.

The dataset contains 32561 tuples and 15 attributes.

Data Preprocessing and Cleaning:

The data is initially pre-processed before conducting situation testing using k-nearest neighbor algorithm. Due to memory constraints and increased computation time, the dataset for this assignment has been sampled to contain 50% of the original data. The sampled data retains the same demographic proportions as the original data, resulting in 16280 tuples and 15 attributes.

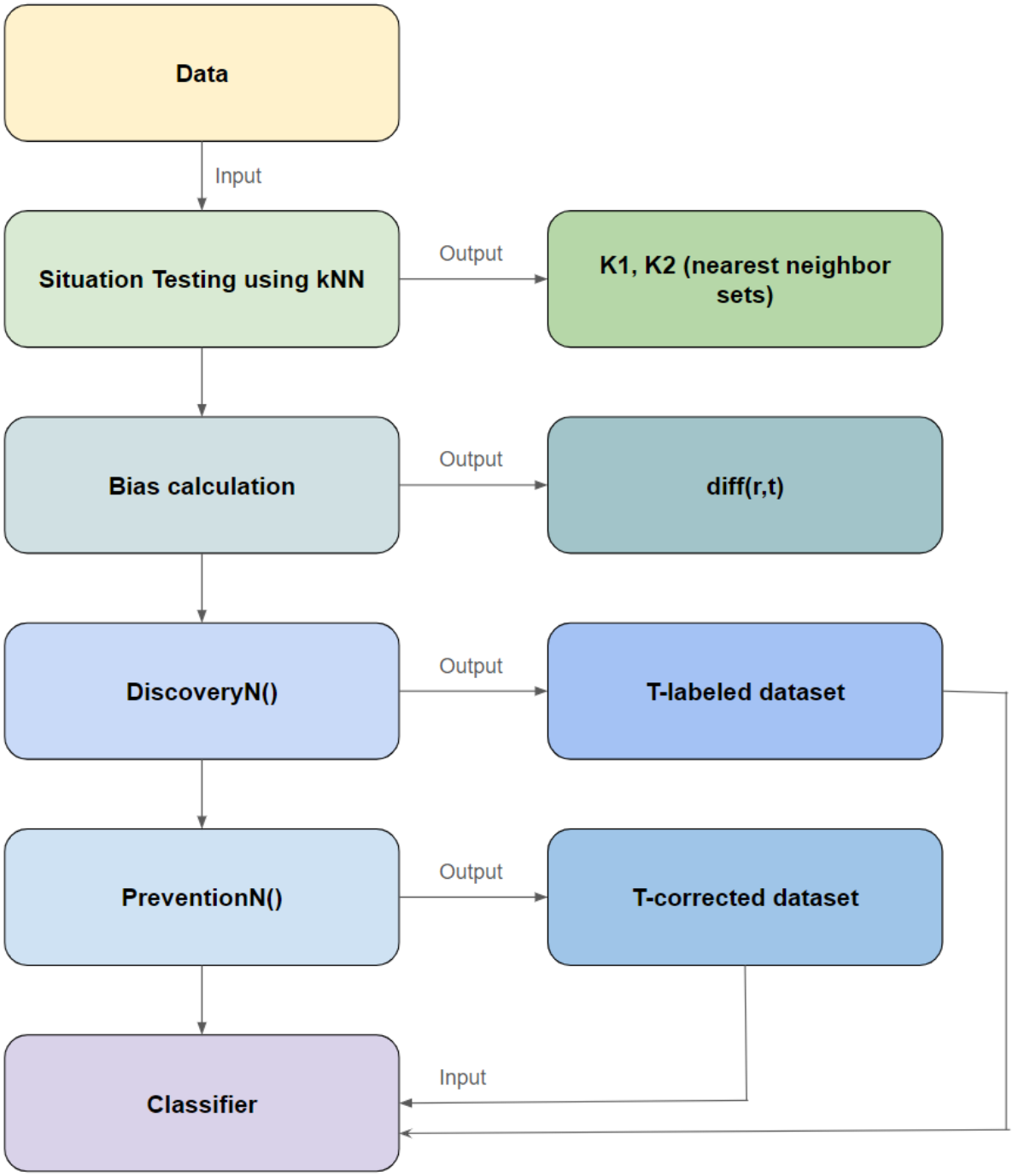
For situation testing using kNN, distance is calculated separately for interval-scaled, nominal and ordinal attributes. All the attributes are divided into the three categories as follows:

Interval-scaled	Age, fnlwgt, capital-gain, capital-loss, hours-per-week
nominal	Workclass, education, occupation, relationship, sex, native-country
ordinal	Education-num

To ensure efficient computation, the nominal attributes are encoded. Nominal attributes such as workclass, occupation and native-country contain missing values '?'. These missing values are retained for distance calculation in kNN algorithm. A penalty has been applied during distance calculation to account for these missing values.

Methodology:

The method used in the assignment follows a structured process to identify bias. This can be simplified into a flowchart as shown below:



Experiments:

Experiment 1: Race as a Sensitive Attribute

The sensitive attribute for this experiment is 'race'. In the dataset, the 'race' attribute takes five values : White, Black, Asian-Pac-Islander, Amer-Indian-Eskimo, Other. Bias is to be identified on income predictions of individuals based on their race. To identify bias, the dataset is split into two groups: protected and unprotected. The protected group contains individuals whose race is not White (i.e., Black, Asian-Pac-Islander, Amer-Indian-Eskimo, Other). The unprotected group contains individuals whose race is White. The goal is to evaluate bias in income predictions based on race.

Experiment 2: Marital Status as a Sensitive Attribute

The sensitive attribute for this experiment is 'marital-status'. In the dataset, the 'marital-status' attribute takes seven values : Married-civ-spouse, Never-married, Divorced, Separated, Widowed, Married-spouse-absent, Married-AF-spouse. In this experiment, the protected group contains individuals whose marital-status is Divorced, Separated or Widowed. The unprotected group contains individuals with marital-statuses of Married-civ-spouse, Never-married, Married-spouse-absent, Married-AF-spouse. The goal here is to identify bias in income predictions based on marital status.

Algorithms:

Situation Testing using k-Nearest Neighbors (kNN):

The bias analysis is performed using the k-nearest neighbors (kNN) algorithm. For each tuple in the dataset, the k-nearest neighbors from both protected and unprotected groups are identified using a distance metric. The metric is computed separately for interval-scaled, nominal, and ordinal attributes. For interval-scaled attributes, the distance between tuples is calculated using the Manhattan distance of z-score normalize values of these attributes. For nominal attributes, distance between categorical values is calculated using the logic: if either value is missing ('?'), a penalty distance of 3 is applied, if values are equal, the distance is set to 0 and if values differ, the distance is set to 1. For ordinal attributes ,the Manhattan distance is applied to scaled values of ordinal attribtues. The pseudo-code of distance metric function are:

```
Function calculate_interval_distance(interval_data1, interval_data2):
    # data1: Interval data from one group (either protected or
    unprotected)
    # data2: Interval data from the other group (or same group for
    intra-group comparison)
    For each pair of tuples in data1 and data2:
        Compute Manhattan distance by summing the absolute differences
        for each interval attribute.
    Return distance_matrix
Function calculate_nominal_distance(nominal_data1, nominal_data2):
    # nominal_data1: Nominal data from one group (either protected or
    unprotected)
    # nominal_data2: Nominal data from the other group (or same group
    for intra-group comparison)
    For each pair of tuples in nominal_data1 and nominal_data2:
        If either value is missing ('?'):
            Set distance to a high penalty value (e.g., 3)
        Else if values are equal:
            Set distance to 0
```

```
            Else:
                  Set distance to 1
      Return distance_matrix
Function calculate_ordinal_distance(ordinal_data1, ordinal_data2):
      # ordinal_data1: Ordinal data from one group (either protected or
      unprotected)
      # ordinal_data2: Ordinal data from the other group (or same group
      for intra-group comparison)
      For each tuple in ordinal_data1 and ordinal_data2:
            Compute Manhattan distance by summing the absolute differences
            for each ordinal attribute
      Return distance_matrix
```

The pseudo-code for the kNN Algorithm is:

```
Function knn(R, k, interval_attr, nominal_attr, ordinal_attr):
      # Step 1: Split the dataset into protected (P_R) and unprotected
      (U_R) groups based on the sensitive attribute.
      P_R = rows in R where 'sensitive attribute' == 'minority'
      U_R = rows in R where 'sensitive attribute' == 'majority'
      # Step 2: Extract the interval, nominal, and ordinal data for both
      groups.
      pr_interval_data = interval_attr data from P_R
      pr_nominal_data = nominal_attr data from P_R
      pr_ordinal_data = ordinal_attr data from P_R

      ur_interval_data = interval_attr data from U_R
      ur_nominal_data = nominal_attr data from U_R
      ur_ordinal_data = ordinal_attr data from U_R
      # Step 3: Calculate the distances between protected group members
      and other members of both protected and unprotected groups.
      # Interval-scaled distance:
      interval_dist_pr = Calculate interval-scaled distance within P_R
      interval_dist_ur = Calculate interval-scaled distance between P_R
      and U_R
      # Nominal distance:
```

```
    nominal_dist_pr = Calculate nominal distance within P_R

    nominal_dist_ur = Calculate nominal distance between P_R and U_R

    # Ordinal distance:

    ordinal_dist_pr = Calculate ordinal distance within P_R

    ordinal_dist_ur = Calculate ordinal distance between P_R and U_R

    # Step 4: Sum the distances to calculate the total distance for both
    protected and unprotected groups.

    total_dist_pr = interval_dist_pr + nominal_dist_pr + ordinal_dist_pr

    total_dist_ur = interval_dist_ur + nominal_dist_ur + ordinal_dist_ur

     # Step 5: Sort the total distances and select the k nearest
    neighbors.

    K1 = Sort total_dist_pr and select the k-nearest neighbors within
    P_R for each tuple in P_R

    K2 = Sort total_dist_ur and select the k-nearest neighbors from U_R
    for each tuple in P_R

    Return K1, K2
```

Bias Calculation:

The bias is calculated by comparing the neighbors' decisions from both the protected (K1) and unprotected groups (K2). The bias is calculated as p1 - p2, where p1 is the proportion of neighbors within protected group with a negative decision and p2 is the proportion of neighbors in unprotected group with a negative decision .

If the bias exceeds a threshold t, it indicates potential unfair treatment.

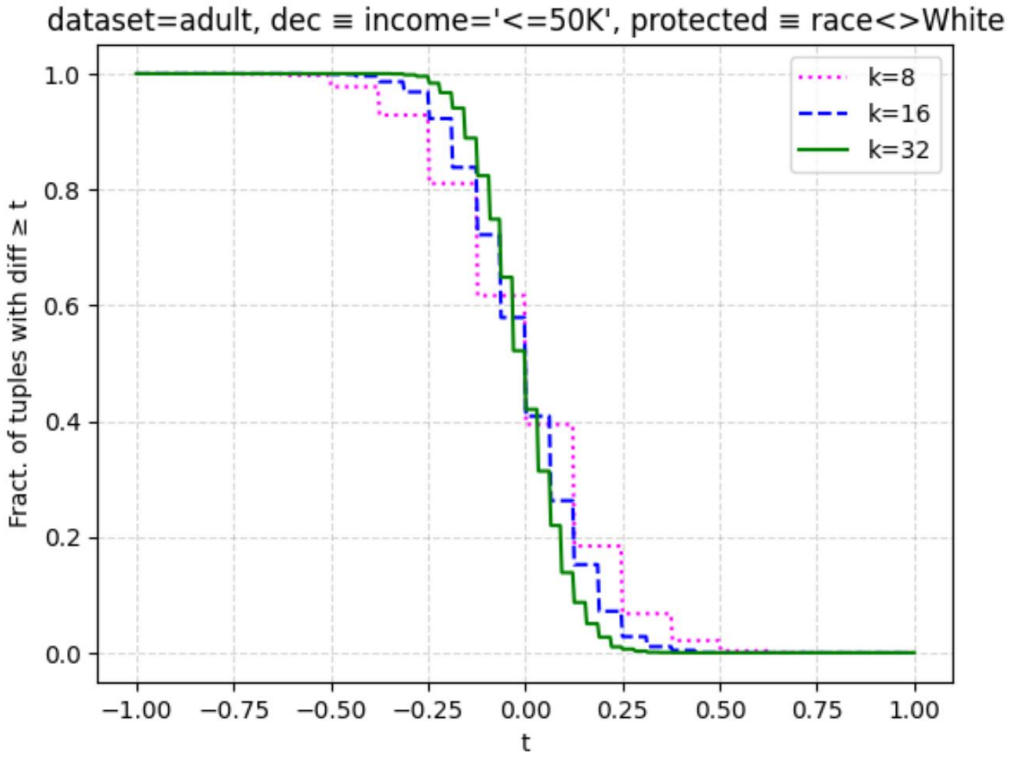
The pseudo-code for the bias calculation is:

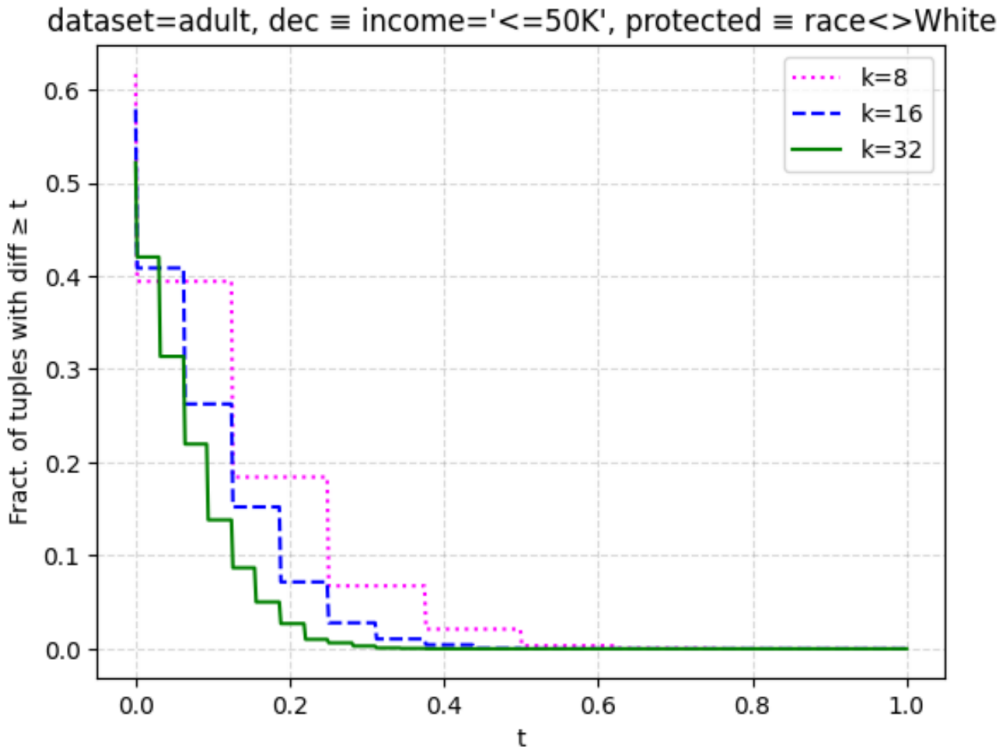
```
Function bias_calculation(K1, K2, P_R):

    # ordinal_data1: Ordinal data from one group (either protected or
    unprotected)

    # ordinal_data2: Ordinal data from the other group (or same group
    for intra-group comparison)

    bias ← []

    For each tuple in P_R(protected group):

        P1 = proportion of neighbors in P_R that have negative
        decision

        P2 = proportion of neighbors in U_R that have positive
        decision

        bias[] ← P1-P2
```

```
Return bias
```

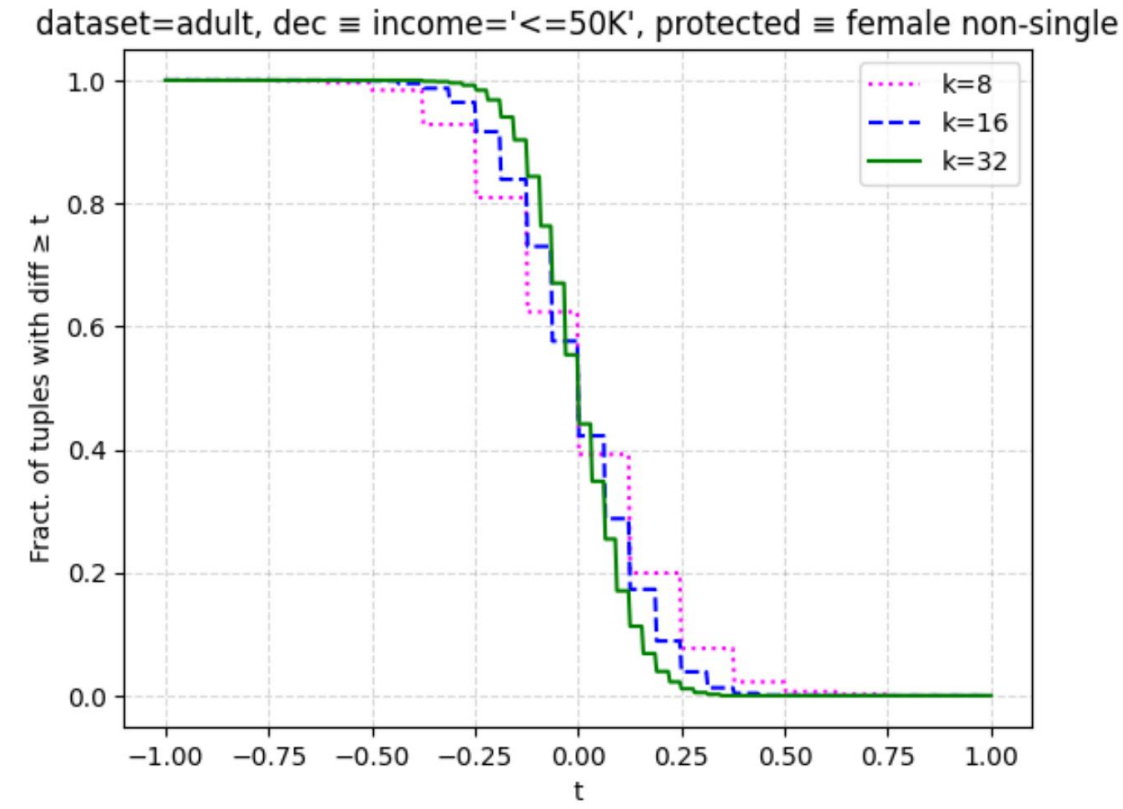
The proportion of individuals in the protected group with a bias greater than or equal to t (threshold of diff/bias) is plotted to visualize the extent of bias.

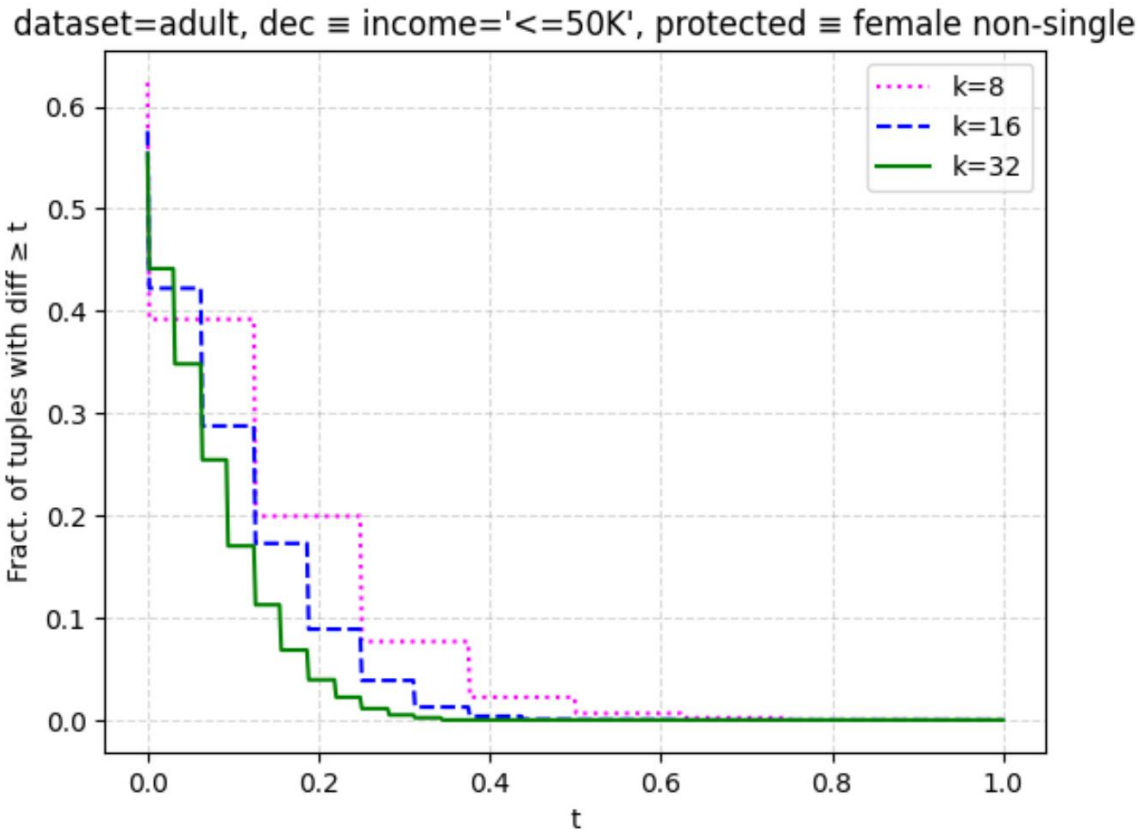
For experiment 1:





For experiment 2:





DiscoverN() Algorithm:

The DiscoverN() algorithm identifies tuples in the protected group where the difference between the protected and unprotected group neighbors decisions exceeds the bias threshold t. The dataset is relabelled as T-labelled.

The T-labelled dataset is evaluated from the DiscoverN() algorithm whose pseudo-code is as follows:

```
Function DiscoverN(Dataset):
    Add a new column disc to the dataset, initialized to 0.
    Iterate through each row of the dataset:
        If the tuple is in protected group, decision is negative and
diff>=t:
                Set the disc column to 1
    Return the modified (t-labelled) dataset.
```

PreventionN () Algorithm:

The PreventionN() algorithm alleviates bias by modifying the decision of tuples in the protected group where discrimination is present (i.e., if the decision is negative and the bias exceeds the threshold). This creates the T-corrected dataset where bias has been adjusted.

The T-corrected dataset is evaluated from the PreventionN() algorithm whose pseudo-code is as follows:

```
Function PreventionN(T-labelled Dataset):
    Iterate through each row of the dataset:
        If the tuple is in protected group, decision is negative and
diff>=t:
                Change the decision to positive
    Return the modified (t-corrected) dataset.
```

Classification Algorithms:

Three classification algorithms were implemented using the scikit-learn library to assess the dataset's performance and fairness: Decision Tree, Naive Bayes, and Logistic Regression classifiers. The classifiers were evaluated on two datasets: T-labelled Dataset (no pre-processing) and T-corrected Dataset (0.10 corrected).

For both datasets, the classifiers were evaluated using the following metrics:

- **Accuracy**: The proportion of correct predictions made by the classifier
- **0.10 Discrimination**: A fairness metric that quantifies the level of discrimination in the predictions.

The results are as follows:

For experiment 1: Race as a Sensitive Attribute

Classifier	No preprocessing		0.10 corrected	
	Accuracy	0.10-discr	Accuracy	0.10-discr.
Decision Tree	96.97 %	0.48 %	98.14 %	1.59 %
Naïve Bayes	31.18 %	15.56 %	82.39 %	3.02 %
Logistic	98.54 %	0.0 %	98.92 %	0.79 %

For experiment 2: Marital Status as a Sensitive Attribute

Classifier	No preprocessing		0.10 corrected	
	Accuracy	0.10-discr	Accuracy	0.10-discr.
Decision Tree	95.08 %	2.62 %	98.89 %	4.28 %
Naïve Bayes	32.25 %	14.51 %	80.67 %	3.45 %
Logistic	96.99 %	0.0 %	98.05 %	2.14 %

The results in the paper 'k-NN as an Implementation of Situation Testing for Discrimination Discovery and Prevention' are:

	No pre-processing		0.10-correction	
classifier	accuracy	0.10-discr.	accuracy	0.10-discr.
C4.5	85.60%	4.24%	84.94%	1.07%
Naïve Bayes	82.46%	4.06%	82.33%	2.23%
Logistic	85.28%	6.61%	84.70%	0.61%
RIPPER	84.42%	5.24%	83.98%	3.94%
PART	85.20%	12.62%	84.00%	2.3%

Observations:

The logistic regression and decision tree models perform better in terms of accuracy than the paper, but the 0.10-discrimination reduction after correction is less. Naive Bayes shows high improvement in accuracy and discrimination in the results compare to the paper. One possible reason why there is no a significant reduction or an increase in 0.10-discrimination after correction could be that the correction was not effectively applied. Also, there could be a difference in how the data is structured after sampling.

Conclusion:

In this assignment, situation testing using the k-nearest neighbor (k-NN) algorithm was implemented to detect discrimination in the adult dataset and DiscoverN and PreventionN algorithms were applied to correct the data.

Three classifiers, namely, Decision Tree, Naïve Bayes, and Logistic Regression, were evaluated in terms of accuracy and 0.10-discrimination before and after bias correction.

The results showed that logistic regression and decision tree models performed well in terms of accuracy, but the 0.10-discrimination reduction after correction have increased. Naïve Bayes, while showing significant improvement in accuracy after correction, still exhibited higher discrimination as compare to the paper's results.