Minimizing Row Removal to Achieve Desired Average Treatment Effect (ATE) in Datasets

Amit Grosman
Technion - Israel Institute of Technology
Under the guidance of assistant professors Brit Youngmann and Shaull Almagor

October 2024

Contents

1 Introduction

The Average Treatment Effect (ATE) is a fundamental metric in causal inference, measuring the mean difference in outcomes between treated and control groups. Accurate estimation of ATE is crucial across various disciplines, including medicine, economics, and public health. However, observational datasets often contain confounders - variables that influence both the treatment assignment and the outcome - resulting in biased ATE estimates.

In order to mitigate this bias, we want to selectively remove data rows to adjust the ATE to a desired range. The primary challenge is to minimize the number of rows removed to preserve the dataset's integrity while achieving precise causal inferences.

This project develops and implements algorithms aimed at efficiently identifying and removing the minimal set of rows necessary to attain a specified ATE range. The methodology progresses from handling datasets with only treatment and outcome variables to more complex scenarios incorporating confounders. The proposed algorithms are evaluated on both synthetic and real-world datasets, demonstrating their effectiveness in balancing data preservation with statistical objectives.

Link to the project's Git repository

2 Problem Definition

2.1 General Description and Formal Definition

The primary objective is to adjust the Average Treatment Effect (ATE) of a dataset to fall within a specified target range by selectively removing data rows.
Formally, given a dataset D comprising n rows, each with a binary treatment indicator T, an outcome variable O, and potentially additional confounders W, the goal is to find a subset $D' \subseteq D$ such that:

$$v - \epsilon \leq \text{ATE}(D') \leq v + \epsilon$$

where:

- v is the target ATE.
- ϵ is the acceptable deviation from the target.

2.2 Calculation of Average Treatment Effect (ATE)

The Average Treatment Effect (ATE) is calculated as the difference in the average outcomes between the treated and control groups. Mathematically, it is defined as:

$$\text{ATE} = \mathbb{E}[O|T = 1] - \mathbb{E}[O|T = 0]$$

where:

- $\mathbb{E}[O|T = 1]$ is the expected outcome for the treated group.
- $\mathbb{E}[O|T = 0]$ is the expected outcome for the control group.

When confounders W are present, they are variables that influence both the treatment assignment T and the outcome O. To account for confounders, the ATE calculation can be adjusted using causal inference techniques to isolate the effect of the treatment from the confounding variables. This ensures that the estimated ATE reflects the true causal impact of the treatment.

The adjusted ATE calculation considering confounders can be expressed as:

$$\text{ATE} = \sum_{w} \left(\mathbb{E}[O|T = 1, W = w] - \mathbb{E}[O|T = 0, W = w]\right) P(W = w)$$

where:

- $\mathbb{E}[O|T = 1, W = w]$ and $\mathbb{E}[O|T = 0, W = w]$ are the expected outcomes for treated and control groups within each confounder value $W = w$.
- $P(W = w)$ is the probability of confounder W having the value w.

3 Proposed Algorithms

3.1 Brute-Force Approach

3.1.1 Description of Brute-Force Method

A brute-force approach to this problem involves evaluating all possible subsets of the dataset D and selecting the one that satisfies the ATE condition with the minimal number of removals. Specifically, for each possible number of rows to remove k (starting from 0 up to n), the algorithm checks every combination of k rows to determine if removing them results in D' with the desired ATE range.

3.1.2 Pseudocode for Brute-Force Method

Algorithm 1 Brute-Force ATE Adjustment

Require: Dataset D with binary attributes T, outcome O, target ATE v, threshold ϵ
Ensure: Minimal number of tuples to remove to achieve $v - \epsilon \leq \text{ATE} \leq v + \epsilon$

```
for k = 0 to n do
    for each subset S ⊆ D with |S| = k do
        D' = D \ S
        Compute ATE(D')
        if v − ε ≤ ATE(D') ≤ v + ε then
            return k
        end if
    end for
end for
return No solution found
```

3.1.3 Why Brute-Force Was Not Utilized

Although the brute-force approach provides an exhaustive search for the minimal number of tuples to remove, its exponential time complexity $O(2^n)$ makes it computationally infeasible for large datasets that are commonly encountered in real-world applications. As the size of the dataset increases, the number of possible subsets grows exponentially, resulting in impractical computation times. Therefore, the focus shifted to more efficient algorithms, which offer significant reductions in computational complexity while still effectively achieving the desired ATE adjustments.

3.2 Naive Removal Algorithm

3.2.1 Description of Naive Method

The **Naive Removal Algorithm** iteratively removes the tuple with the highest or lowest outcome O from the treated or control group, respectively, until the ATE falls within the desired range. This straightforward approach focuses on getting to the desired range as fast as possible, but in the price of not being able to fine tune the removal process when being close the the desired range. While simple and easy to implement, the naive method does not always achieve the most optimal removal of tuples.

3.3 Binary Search Removal Algorithm

3.3.1 Description of Binary Search Method

The **Binary Search Removal Algorithm** enhances the naive approach by incorporating a binary search strategy to identify the optimal row for removal every iteration. Instead of solely removing the highest or lowest O values, this method simulates the removal of candidates to determine which row most effectively brings the ATE closer to the target. By evaluating the impact of potential removals, the binary search method achieves greater precision in adjusting the ATE, but at the cost of increased computational effort compared to the naive method.

3.4 Greedy Combined Algorithm

3.4.1 Description of Greedy Combined Algorithm

The **Greedy Combined Algorithm** derives its name from the integration of the **Naive Removal Algorithm** and the **Binary Search Removal Algorithm**. Initially, it employs the naive strategy of removing the highest or lowest O values to adjust the ATE. Upon detecting a sign change in the difference between the current ATE and the target (indicating that the removal has overshot the desired range), the algorithm transitions to a binary search approach. This hybrid strategy leverages the speed of the naive method and the precision of the binary search method, aiming to achieve optimal ATE adjustments efficiently.

3.4.2 Pseudocode

Below is the pseudocode for the Greedy Combined Algorithm:

Algorithm 2 Greedy Combined Algorithm

Require: Dataset D with a binary attribute T and outcome O, a target difference v and a threshold ϵ
Ensure: The number of tuples deleted from D
  /* Initialize the groups, their sizes and current difference */
  $D0$, $D1$, $v_c \leftarrow$ InitializeGroups(D, T, O)
  num_removals $\leftarrow 0$
  prev_diff $\leftarrow v_c - v$
  switched $\leftarrow$ False
  /* Greedy naive approach loop */
  while $v_c \notin [v-\epsilon, v+\epsilon]$ **and** $D0$, $D1$ not empty **do**
    if $v_c > v + \epsilon$ **then**
      Remove tuple with max O from $D1$
    else if $v_c < v - \epsilon$ **then**
      Remove tuple with max O from $D0$
    end if
    Update $v_c \leftarrow$ UpdateDifference($D0$, $D1$)
    num_removals $\leftarrow$ num_removals + 1
    curr_diff $\leftarrow v_c - v$
    if prev_diff $\times$ curr_diff < 0 **then**
      switched $\leftarrow$ True
      break
    end if
    prev_diff $\leftarrow$ curr_diff
  end while
  /* Switch to binary search for fine-tuning */
  if switched **then**
    while $v_c \notin [v-\epsilon, v+\epsilon]$ **and** $D0$, $D1$ not empty **do**
      best_candidate $\leftarrow$ BinarySearch(candidates, v)
      Remove *best_candidate*
      Update $v_c \leftarrow$ UpdateDifference($D0$, $D1$)
      num_removals $\leftarrow$ num_removals + 1
    end while
  end if
  return num_removals

3.4.3 Algorithm Explanation

1. **Initialization**:
   - **Groups Setup**: Divide the dataset D into treated $D1$ and control $D0$ groups based on the treatment indicator T.
   - **Aggregates Calculation**: Compute initial sums and sizes for each group and the current ATE difference v_c.
   - **Counters Initialization**: Initialize a counter to track the number of removals.
   - **Difference Tracking**: Initialize variables to monitor the difference relative to the target v.
2. **Greedy Removal Loop**:
   - **Row Removal Decision**:
     - If $v_c > v + \epsilon$: Remove the tuple with the maximum O from the treated group $D1$.

- If $v_c < v - \epsilon$: Remove the tuple with the maximum O from the control group $D0$.

- **Aggregate Update**: After removal, update the sums and counts for the respective groups.
- **Difference Recalculation**: Recompute the ATE difference v_c.
- **Counter Increment**: Increment the removal counter.
- **Switching Check**: If the product of the previous difference and the current difference is negative (prev_diff $\times$ curr_diff < 0), a sign change has occurred, indicating that the algorithm has overshot the desired range. This triggers a switch to the binary search phase for fine-tuning.

3. **Switch to Binary Search for Fine-Tuning**:

   - **Binary Search Loop**:
     - Continue removing the best candidate identified through binary search until the ATE falls within the desired range or no further removals are possible.
   - **Row Removal**: Identify and remove the best candidate row that brings the ATE closest to the target.
   - **ATE Update**: Recalculate the ATE using the updated dataset.
   - **Counter Increment**: Increment the removal counter.

4. **Termination**:

   - The algorithm terminates when the ATE difference v_c is within the desired range or when no further removals can be made without violating group sizes.

3.5 Per Group Removal Algorithm with Confounders

3.5.1 Description of Per Group Removal Algorithm

To handle datasets with confounders, we extend the Greedy Combined Algorithm to the **Per Group Removal Algorithm**. This method ensures that row removals are balanced across different confounder values. Confounder groups refer to the unique combinations of treatment and confounder values, enabling targeted and balanced adjustments.

3.5.2 Pseudocode

Below is the pseudocode for the Per Group Removal Algorithm:

Algorithm 3 Per Group Removal Algorithm

Require: Dataset D with binary attributes T, Confounders $W1$, $W2$, ... , and outcome O, target difference v, threshold ϵ

Ensure: Number of tuples deleted from D

```
/* Initialize */
num_removals ← 0
sorted_indices ← Sort rows by O for each (T, Confounder Combination) group
current_ATE ← CalculateATE(D)
/* Removal loop */
while current_ATE ∉ [v − ε, v + ε] and rows exist in groups do
    if current_ATE > v + ε then
        candidates ← Select highest O from each group
    else
        candidates ← Select lowest O from each group
    end if
    best_diff ← ∞
    row_to_remove ← None
    for each candidate in candidates do
        new_ATE ← SimulateRemove(D, candidate)
        diff ← |new_ATE − v|
        if diff < best_diff then
            best_diff ← diff
            row_to_remove ← candidate
        end if
    end for
    if row_to_remove is None then
        break
    end if
    RemoveTuple(D, row_to_remove)
    num_removals ← num_removals + 1
    current_ATE ← CalculateATE(D)
end while
return num_removals
```

3.5.3 Algorithm Explanation

1. **Initialization**:
    - **Counters Initialization**: Initialize a counter to track the total number of removals.
    - **Sorted Indices**: Sort the dataset rows by the outcome variable O within each treatment T and confounder W group. Binning may be used to reduce the number of confounder combinations, thereby lowering computational costs.
    - **Initial ATE Calculation**: Compute the initial ATE using the full dataset.
2. **Main Removal Loop**:
    - **Row Removal Decision**:
        - If $current\_ATE > v + \epsilon$: Select the row with the highest O from each confounder group.
        - Else: Select the row with the lowest O from each confounder group.
    - **Best Candidate Identification**: Among all candidates, determine the row whose removal results in the smallest difference from the target ATE.

- **Row Removal**: Remove the identified row from the dataset and increment the removal counter.
- **ATE Update**: Recalculate the ATE using the updated dataset.
- **Termination Check**: If the ATE falls within the desired range, terminate the loop.

4 Experiments

This section presents three experiments designed to evaluate the effectiveness of the proposed algorithms in minimizing row removals to achieve the desired Average Treatment Effect (ATE) range. The experiments are categorized as follows:

1. **Synthetic Data Experiments:** The first two experiments utilize synthetic datasets to assess the performance of the Naive Removal Algorithm, Binary Search Removal Algorithm, Greedy Combined Algorithm, and Per Group Removal Algorithm under controlled conditions.

2. **Real-World Data Experiment:** The third experiment applies the Per Group Removal Algorithm to a real-world dataset to demonstrate its practical applicability and effectiveness in adjusting the ATE in complex, real-life scenarios.

4.1 Explanation of Epsilon and Epsilon Factor

In the context of these experiments, two related but distinct parameters are used to control the precision of ATE adjustments: **epsilon** and **epsilon factor**.

- **Epsilon (ϵ):** This is the allowed deviation from the target ATE v. It defines the acceptable range within which the adjusted ATE must fall:

$$v - \epsilon \leq \text{ATE} \leq v + \epsilon$$

- **Epsilon Factor (ϕ):** This is a parameter used in the synthetic data experiments to dynamically set the value of ϵ. The relationship between ϵ and ϕ is defined as:

$$\epsilon = \frac{|\text{Original ATE} - \text{Desired ATE}|}{\phi}$$

  As seen in the equation, higher epsilon factor ϕ results in a smaller ϵ, demanding more precise adjustments to the ATE. This dynamic setting allows the experiments to explore different levels of precision and their impact on algorithm performance.

4.2 Experiment Configuration

4.2.1 First Experiment: Synthetic Data without Confounders

- **Objective**: Evaluate the performance of the Naive Removal Algorithm, Binary Search Removal Algorithm, and Greedy Combined Algorithm in adjusting the ATE without the influence of confounders.

- **Database Sizes**: $\{1{,}000{,}000; 2{,}000{,}000; 3{,}000{,}000; \ldots; 10{,}000{,}000\}$

- **Number of Trials**: 3 per configuration.

- **Epsilon Adjustment Factors** (ϕ): $\{10; 100; 1{,}000; 10{,}000; 100{,}000\}$

- **Row Removal Strategy**: Rows removed at random were from the whole dataset.

4.2.2 Second Experiment: Synthetic Data with Binary Confounder W

- **Objective**: Assess the effectiveness of the Per Group Removal Algorithm in handling datasets with a binary confounder W, ensuring balanced row removals across confounder groups.
- **Database Sizes**: $\{1{,}000{,}000; 2{,}000{,}000; 3{,}000{,}000; \ldots; 10{,}000{,}000\}$
- **Number of Trials**: 3 per configuration.
- **Epsilon Adjustment Factors (ϕ)**: $\{10; 100; 1{,}000; 10{,}000\}$
- **Row Removal Strategy**: Rows removed at random were exclusively from $W = 0$, while tracking the number of rows removed from both $W = 0$ and $W = 1$.

4.2.3 Third Experiment: Real-World Data Application

- **Database**: StackOverflow Survey Dataset (available in the Git repository)
- **Database Size**: 38,090 records
- **Required ATE Modification**: Increase by $500
- **Row Removal Strategy**: The Per Group Removal Algorithm was applied to selectively remove the lowest-paid individuals from appropriate confounder groups to achieve the desired ATE increase with minimal row removal.

4.3 Experimental Procedure

4.3.1 Synthetic Data Experiments (First and Second Experiments)

For each combination of dataset size and epsilon adjustment factor (ϕ) in the first two experiments, the following steps were performed:

1. **Dataset Generation**: Synthetic datasets were generated with predefined treatment assignments and outcomes. In the second experiment, a binary confounder W was added to introduce complexity and simulate real-world data conditions.
2. **Row Removal Simulation**: Initially, 10% of rows were removed at random (from $W = 0$ in the second experiment) to set the desired ATE baseline.
3. **Epsilon Calculation**: Using the epsilon factor ϕ, the allowed deviation ϵ was calculated to define the target ATE range.
4. **Algorithm Execution**: The respective algorithms - Greedy Combined Algorithm, Naive Removal Algorithm, and Binary Search Removal Algorithm for the first experiment, and Per Group Removal Algorithm for the second experiment - were executed to adjust the ATE within the specified range.
5. **Measurement**: The number of tuples removed and the time taken for each algorithm were recorded to assess efficiency and effectiveness.
6. **Averaging**: Results were averaged over the three trials for each configuration to account for variability and ensure reliability of the outcomes.

4.3.2 Real-World Data Experiment (Third Experiment)

In the third experiment, the focus shifted to applying the Per Group Removal Algorithm on real-world data to validate its practical utility.

Experiment Setup:

- **Treatment Variable** T: FormalEducation (Yes/No).
- **Outcome Variable** O: ConvertedSalary.
- **Confounders**: UndergradMajor, Continent, RaceEthnicity.
- **Desired ATE Range**: The ATE was set to increase by \$500 from its original value, with $\epsilon = 1$ to allow a minimal margin of error.
- **Row Removal Strategy**: To increase the ATE as required, the algorithm selectively removed the lowest-paid individuals from the treated groups within each confounder combination. This targeted removal ensures that the ATE increases while minimizing the total number of rows removed.

The procedure was as follows:

1. **Dataset Preparation**: Loaded the StackOverflow Survey Dataset comprising 38,090 records, ensuring the relevant attributes - FormalEducation, UndergradMajor, Continent, RaceEthnicity, and ConvertedSalary - were appropriately formatted for analysis.
2. **ATE Modification Goal**: Set the objective to increase the ATE of ConvertedSalary by \$500. Unlike the synthetic experiments, the ATE modification was intentional rather than random.
3. **Row Removal Strategy**: Applied the Per Group Removal Algorithm to selectively remove the lowest-paid individuals from specific confounder groups. This targeted approach aimed to increase the ATE effectively while minimizing the total number of rows removed.
4. **Algorithm Execution**: Executed the Per Group Removal Algorithm to adjust the ATE within the desired range.
5. **Measurement**: Recorded the number of tuples removed, the execution time, and the impact on each confounder group to evaluate the algorithm's performance in a real-world context.
6. **Result Analysis**: Analyzed the progression of the ATE towards the desired increase, ensuring that the modifications were meaningful and maintained the dataset's integrity.

4.4 Summary

By conducting these three experiments, the project evaluates the proposed algorithms capabilities in both synthetic environments and real-world datasets. The first two experiments establish the effectiveness and efficiency of the algorithms under varying conditions, while the third experiment demonstrates their practical use in achieving meaningful ATE adjustments in real-world scenarios.

5 Results

This section presents the results of the experiments conducted to evaluate the performance of the algorithms across different dataset sizes and epsilon factors.

5.1 First Experiment: Without Confounders

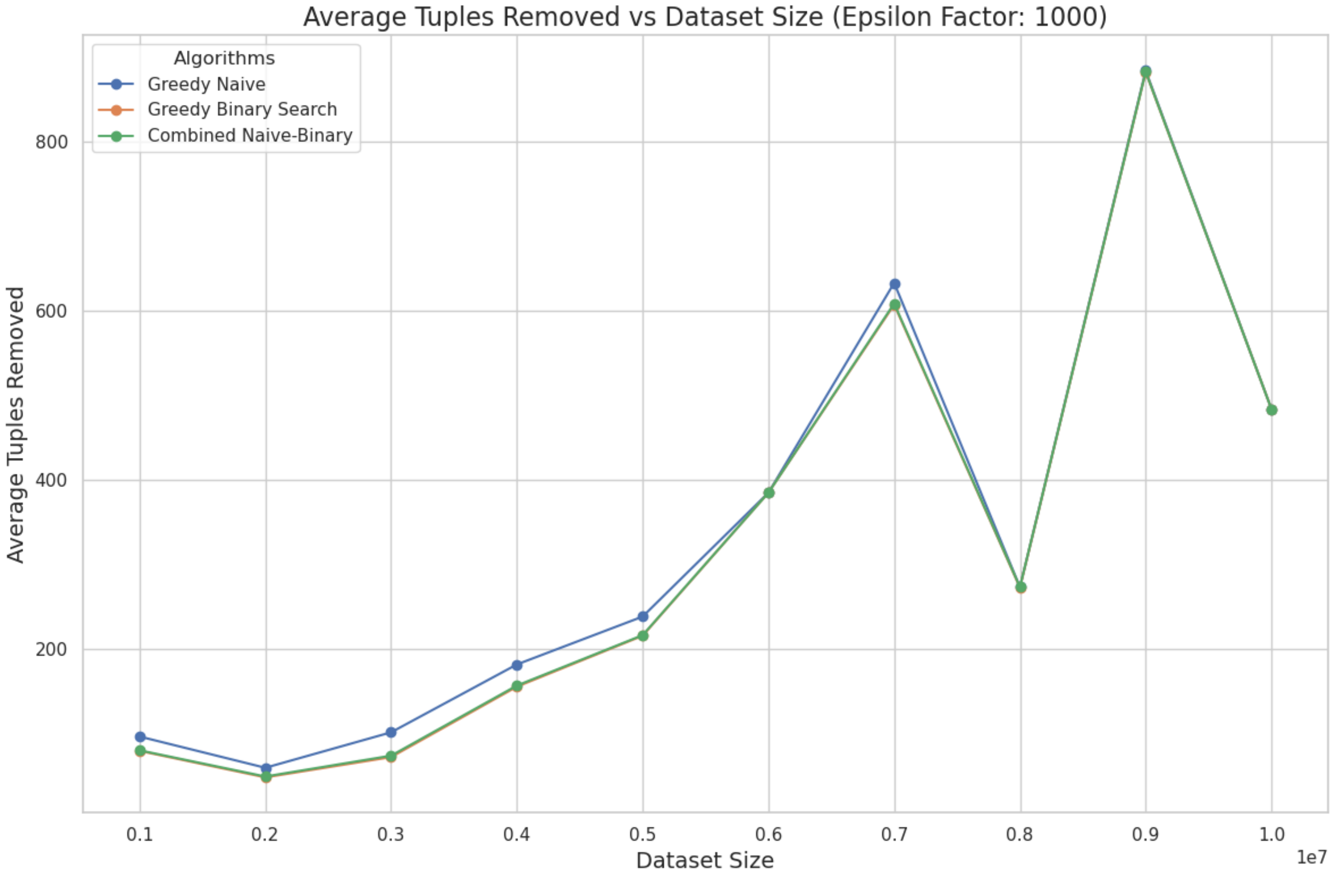

Figure 1: Average Tuples Removed vs. Dataset Size for Epsilon Factor = 1,000 (No Confounder)

This plot illustrates the average number of tuples removed in the first experiment when the epsilon factor is set to 1,000. The Greedy Combined Algorithm and the Binary Search Removal Algorithm consistently removed a similar number of tuples, closely followed by the Naive Removal Algorithm, which performed slightly behind. However, the differences among the three algorithms were negligible, indicating that the combined approach effectively matches the precision of the binary search method while maintaining the efficiency of the naive method.

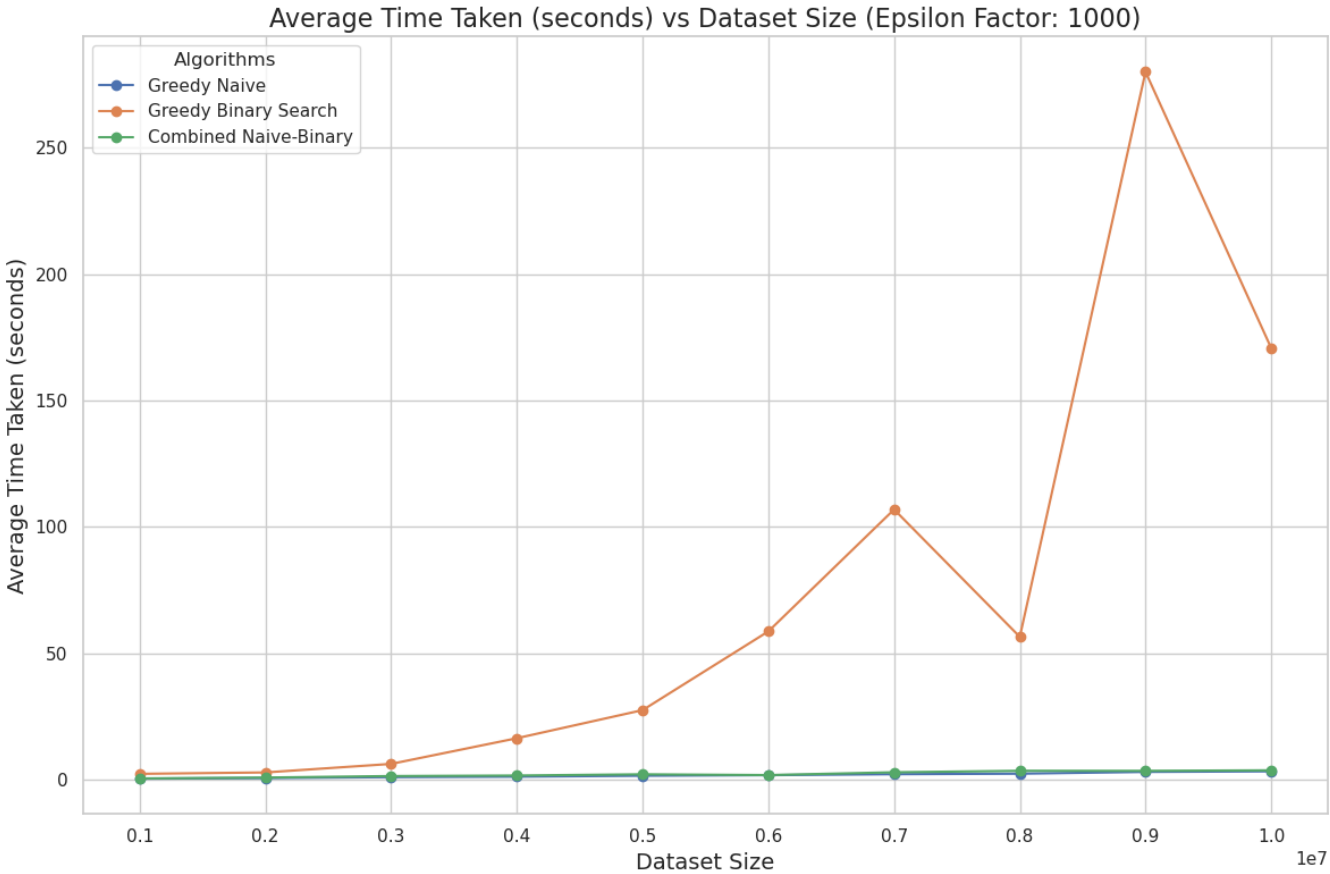

Figure 2: Average Time Taken vs. Dataset Size for Epsilon Factor = 1,000 (No Confounder)

This plot shows the average time taken by the algorithms in order to perform the ATE adjustment across various dataset sizes with an epsilon factor of 1,000. The Naive Removal Algorithm and the Greedy Combined Algorithm demonstrated significantly faster execution times compared to the Binary Search Removal Algorithm, which required more computational time due to its intensive search process.

5.1.1 Summary

Across all dataset sizes ranging from 1,000,000 to 10,000,000, the Greedy Combined Algorithm consistently removed a similar number of tuples as the Binary Search Removal Algorithm, while matching the Naive Removal Algorithm in terms of computational speed. This indicates that the combined approach effectively balances the precision of causal adjustments with operational efficiency, making it a superior choice for large-scale applications.

5.2 Second Experiment: With Binary Confounder W

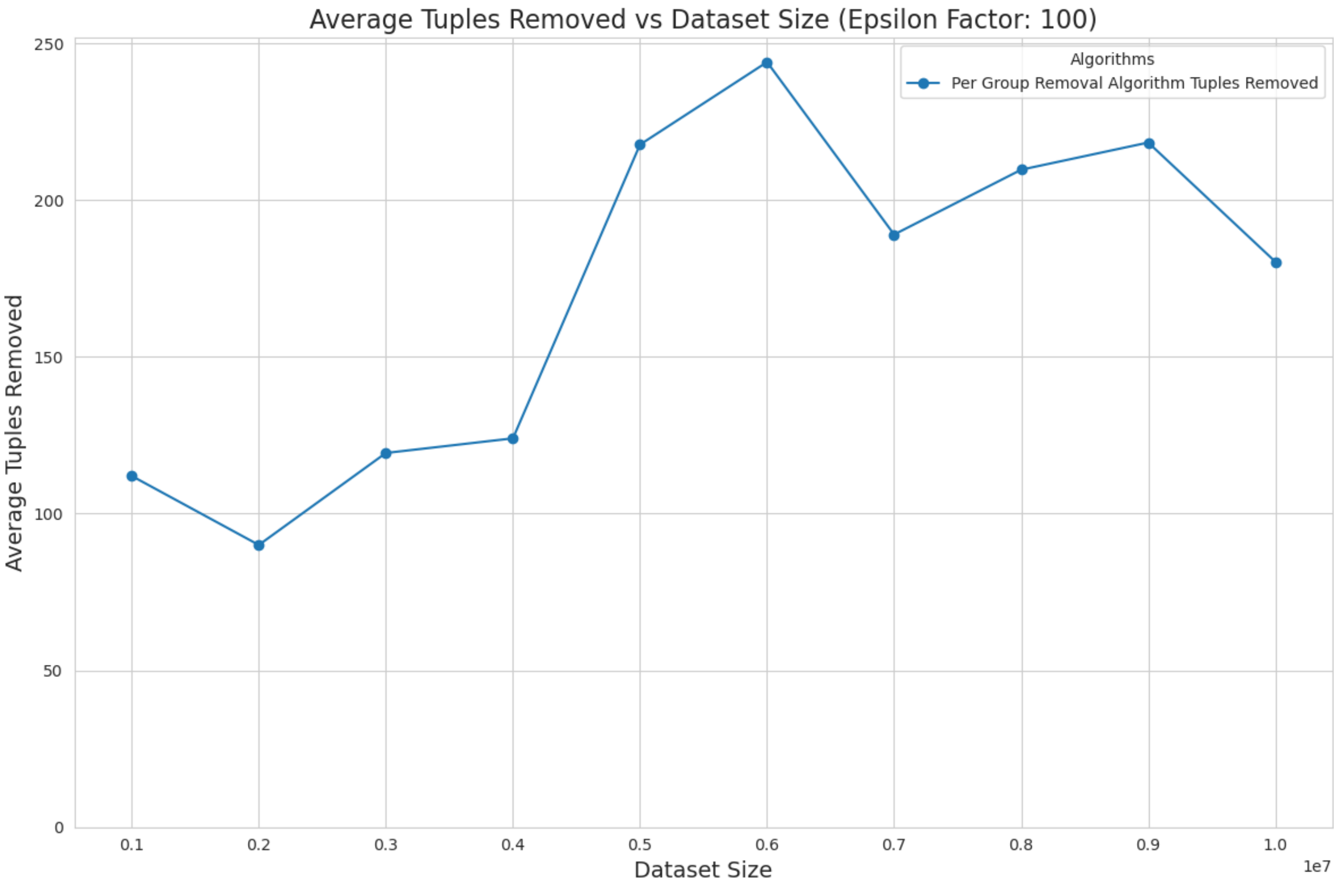

Figure 3: Average Tuples Removed vs. Dataset Size for Epsilon Factor = 100 (Per Group Removal)

This plot shows the average number of tuples removed by the Per Group Removal Algorithm across different dataset sizes when the epsilon factor is 100. The algorithm maintained a consistent number of tuples removed, ranging from 100 to 250. This consistency indicates the algorithm's effectiveness in managing confounders without being significantly affected by the scale of the dataset.

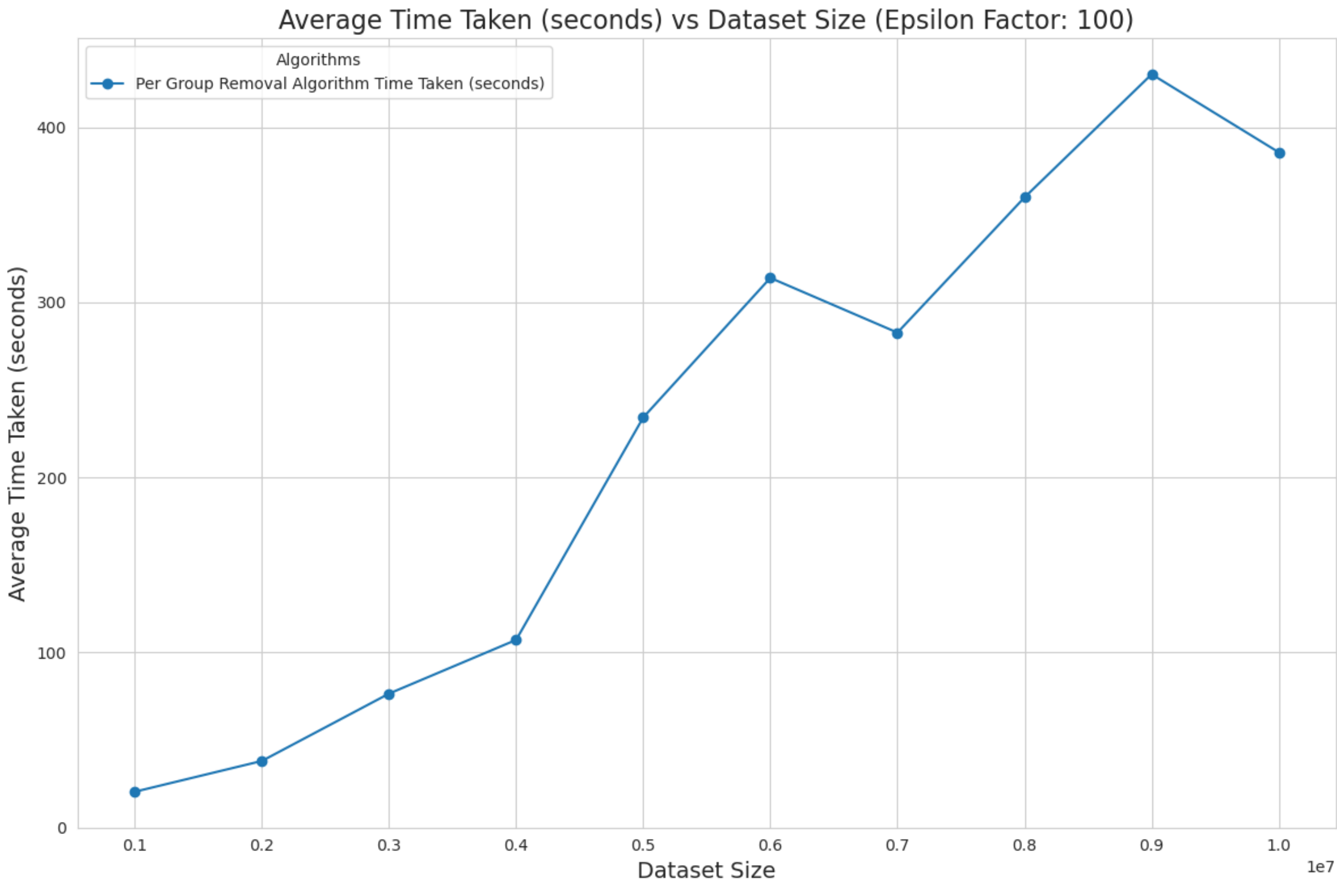

Figure 4: Average Time Taken vs. Dataset Size for Epsilon Factor = 100 (Per Group Removal)

This plot presents the average time taken by the Per Group Removal Algorithm to adjust the ATE across various dataset sizes with an epsilon factor of 100. The results show an almost linear growth with increasing dataset sizes. This behavior is expected, as larger datasets require more computational effort to process, especially when handling multiple confounder groups.

5.2.1 Summary

In the presence of the binary confounder W, the Per Group Removal Algorithm effectively removed rows from both $W = 0$ and $W = 1$ groups to achieve the desired ATE. Specifically, for an epsilon factor of 100, the algorithm removed a consistent number of tuples across all dataset sizes, maintaining dataset integrity while aligning the ATE within the specified range. The time taken by the algorithm scaled almost linearly with dataset size, demonstrating its scalability and efficiency in managing confounders.

5.3 Third Experiment: Real-World Data

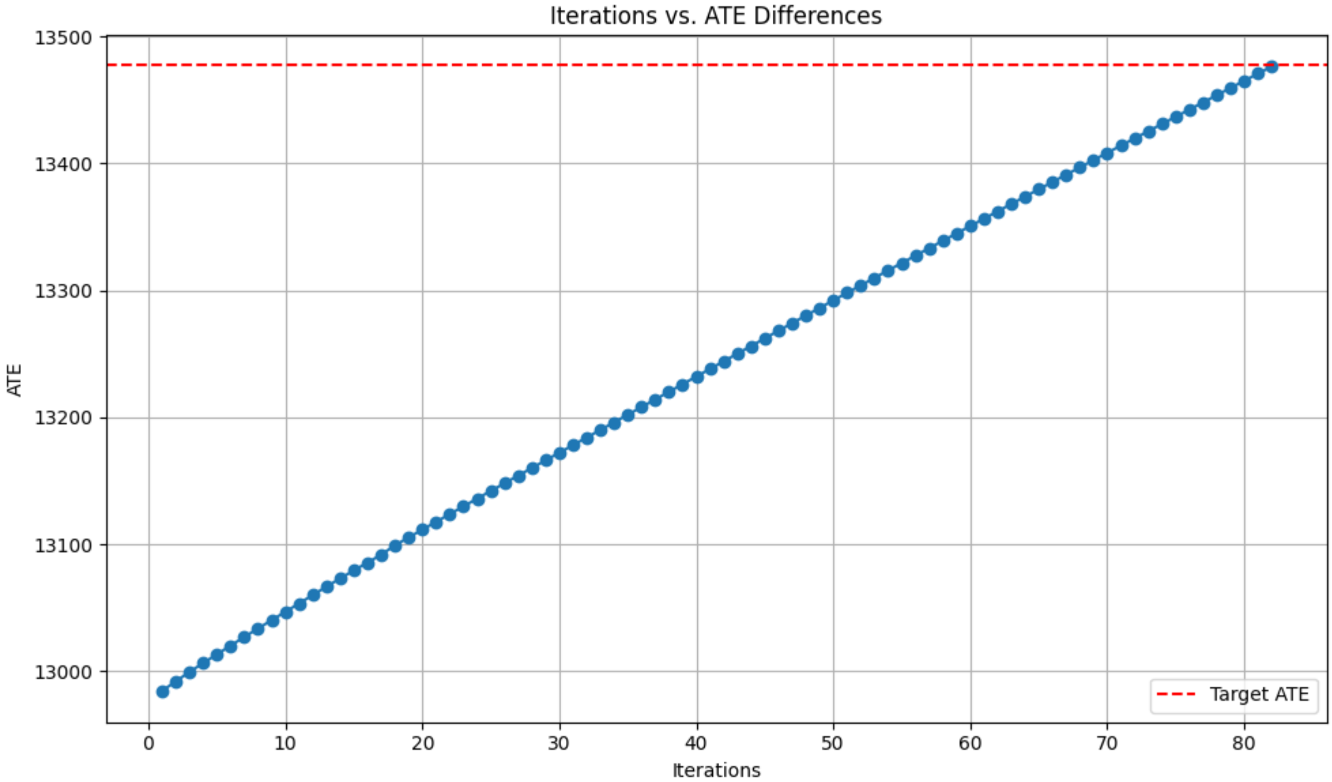

Figure 5: ATE Progression Over Iterations for Real-World Dataset

This plot illustrates the progression of the ATE as the Per Group Removal Algorithm iteratively removed rows from the StackOverflow survey dataset. Each iteration represents a row removal, and the corresponding ATE is plotted to show how it approaches the desired range. The plot demonstrates a steady convergence towards the target ATE, highlighting the algorithm's effectiveness in adjusting the ATE with minimal data loss.

5.3.1 Summary

In the third experiment, the Per Group Removal Algorithm was successfully applied to a real-world StackOverflow survey dataset containing 38,090 records. The algorithm aimed to increase the ATE of ConvertedSalary by $500 within a tight epsilon margin of 1. To achieve this, the algorithm strategically removed the lowest-paid individuals from specific confounder groups, ensuring that the ATE increased while minimizing the total number of rows removed.

Key findings from the experiment include:

- **Total Removals**: 82 tuples were removed to achieve the desired ATE increase.
- **Average Salary of Removed Rows**: $21,332.10 per year, indicating that the algorithm targeted lower-income individuals to effectively raise the ATE.
- **Groups Affected**: The majority of removals occurred in groups with Formal Education as "Yes," UndergradMajors in "Engineering & Math" and "Other Disciplines," Continent as "North America" and "OC," and RaceEthnicity as "Asian Descent" and "European Descent."
- **Execution Time**: The algorithm completed the ATE adjustment in approximately 145 seconds, demonstrating reasonable computational efficiency given the dataset size.

- **ATE Progression**: The ATE steadily increased towards the target range with each row removal, as depicted in Figure 5, showcasing the algorithm's effectiveness in real-world scenarios.

The experiment shows that the Per Group Removal Algorithm can adjust the ATE in real-world datasets by removing the least impactful rows, thereby preserving data integrity and achieving the desired causal inference with minimal data loss.

6 Conclusion

This project developed and implemented algorithms for minimizing row removals while achieving a desired Average Treatment Effect (ATE) in datasets. The Greedy Combined Algorithm demonstrated a balance between accuracy and efficiency, offering a solution that performs well even in large datasets. It combines the strengths of both the Binary Search and Naive Removal approaches. The Per Group Removal Algorithm showed it is capable of handling confounders, preserving the dataset's structure. The algorithm's application to real-world data further demonstrates its practicality in real-world scenarios, making it a usable tool for data analysis and ATE adjustments.