SUPPLEMENTARY MATERIALS
Fair Wasserstein Coresets

A Experiment Details

A.1 Runtime Analysis on Synthetic Dataset

As mentioned in Section 5, we generate a synthetic dataset in which one feature is strongly correlated with the protected variable D to induce a backdoor dependency on the outcome. We consider a binary protected variable, $D \in \{0, 1\}$, which could indicate e.g., gender or race. The synthetic dataset contains two features, a feature X_1 correlated with the protected variable and a feature X_2 uncorrelated with the protected variable. For $D = 0$, X_1 is uniformly distributed in $[0, 10]$, while for $D = 1$, $X_1 = 0$. Instead, X_2 is 5 times a random variable from a normal distribution $\mathcal{N}(0, 1)$. Finally, the outcome Y is binary, so $Y = \{0, 1\}$: $Y_i = 1$ when $Y_i > m_x + \epsilon_i$ and $Y_i = 0$ when $Y_i \leq m_x + \epsilon_i$, where m_x is the mean of $\{(X_1)_i + (X_2)_i\}_i$ and the noise ϵ_i comes from a normal distribution $\mathcal{N}(0, 1)$.

This experiment visualizes the speed of our method with respect to different numbers of overall samples n, number of samples in the compressed dataset m, and the dimensionality of features p. We evaluated the performance of the algorithm under the synthetic data with different configurations of n, p, and m. In this experiment, we set compute the fair Wasserstein coreset under the l_1-norm distance and we use the k-means to initialize the starting coreset $\widehat{X}^0$. We terminate the algorithm when $\widehat{X}^k = \widehat{X}^{k-1}$. The time per iteration and total iterations for varying n, p, and m are shown in the Figures 1 (top left) and 2. We see that increasing the sample size of the original dataset n increases the runtime and number of iterations (keeping them within a maximum of 5 minutes), while increasing the number of coresets m or dimensionality of the features p reduces the overall numbers of iterations but increases each iteration's runtime.

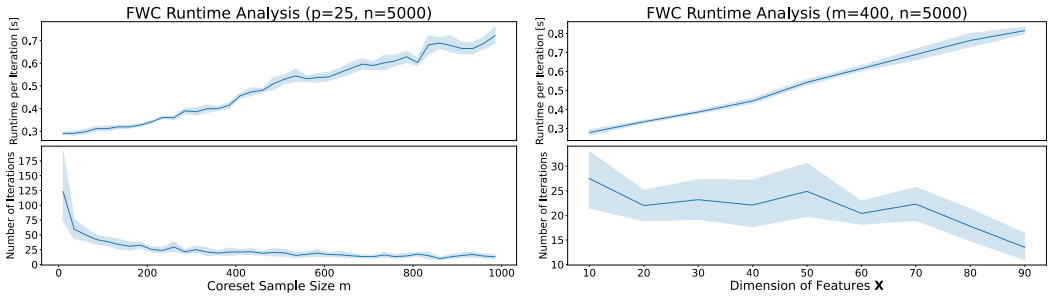

Figure 2: Runtime analysis of `FWC` when varying the size of the coreset m (left) and the dimensionality of the features p (right). We report averages and one standard deviation computed over 10 runs.

A.2 Real Datasets

We consider the following four real datasets widely used in the fairness literature [13]:

- the *Adult dataset* [3], which reports demographic data from the 1994 US demographic survey about $\sim 49,000$ individuals. We use all the available features for classification, including gender as the protected variable D and whether the individual salary is more than $50,000$USD;
- the *Drug dataset* [14], which contains drug usage history for 1885 individuals. Features X include the individual age, country of origin, education and scores on various psychological test. We use the individual gender as the protected variable D. The response Y is based on whether the individual has reported to have never used the drug "cannabis" or not;

- the *Communities and Crime dataset* [33] was created towards the creation of a software tool for the US police department. The dataset contains socio-economic factors for $\sim 2,000$ communities in the US, along with the proportion of violent crimes in each community. As protected variable D, we include whether the percentage of the black population in the community is above the overall median. For the response Y, we use a binary indicator of whether the violent crimes percentage level is above the mean across all communities in the dataset;
- the *German Credit dataset* [17] reports a set of $1,000$ algorithmic credit decisions from a regional bank in Germany. We use all the available features, including gender as protected variable D and whether the credit was approved as response Y.

We also include the implementation and hyper-parameters of the methods used in the fairness-performance tradeoffs presented in Figures 1 and 5

- For `FWC` we set the hyper-parameter ϵ of problem in Equation (3.4) to be $\epsilon = [0.01, 0.05, 0.1]$, hence obtaining three separate `FWC` models, `FWC` (0.01), `FWC` (0.05) and `FWC` (0.1);
- For `Fairlet` [2], we use the implementation available at the following GitHub repository: `https://github.com/talwagner/fair_clustering/tree/master`
- For `IndFair` [7] and `K-Median Coresets` [1], we use the implementation available at the following GitHub repository: `https://github.com/jayeshchoudhari/CoresetIndividualFairness/tree/master`
- For k-means [25] and k-medoids [27, 30] we use the implementations available in the Python package `Scikit-Learn` [31]

All computations are run on an Ubuntu machine with 32GB of RAM and 2.50GHz Intel(R) Xeon(R) Platinum 8259CL CPU. For all datasets, we randomly split 75% of the data into training/test set, and change the split during each separate run; the training data are further separated into training and validation with $90/10$ to compute early stopping criteria during training. The downstream classifier used is a one-layer deep multi-layer perceptron with 20 hidden layers, ReLu activation function in the hidden layer and softmax activation function in the final layer. The learning rate is set to 10^{-3}, with a batch size of 32, and a maximum number of epochs set to 500 with early stopping evaluated on the separate validation set with a patience of 10 epochs.

Table 2 and Figures 3 and 4 include the ranking and numerical results for all methods in terms of Wasserstein distance from the original dataset and clustering cost, for the three coreset sizes $m = [5\%, 10\%, 20\%]$ (apart from the Adult dataset, in which coreset sizes are set to $m = [0.5\%, 1\%, 2\%]$ due to the large size of the original dataset). Clustering cost is computed as the sum of the squared distance of each points in the original dataset from the closest coreset representative, while the Wasserstein distance is computed solving the optimal transport between the empirical distribution of the original dataset and the one of the coresets. `FWC` provides the closest distance in Wasserstein distance almost uniformly, with the only exception of the Credit dataset, in which the large number of discrete features makes the optimization non-smooth in feature space, resulting in a potentially imprecise solution of Equation (3.9). In addition, `FWC`, while not naturally targeting clustering costs, seems to achieve competitive clustering costs in smaller datasets.

Figures 1 and 5 show how `FWC` achieves a competitive fairness-utility tradeoff with respect to all other methods, by consistently reducing disparity in the downstream classification with respect to other approaches, and often maintaining the same level of accuracy. For completeness, Figure 6 also reports the fairness-utility tradeoff performances with 1σ (one standard deviation lines); some of the standard deviations are very large due to the MLP classifier becoming trivial (i.e., always returning 0s or 1s), which yields very low performance but has no demographic disparity (by definition, as all test samples are assigned the same outcome). Finally, note that due to the size of the Adult dataset, the `Fairlet` [2] could not run due to the RAM memory required exceeding the machine capacity (32GB).

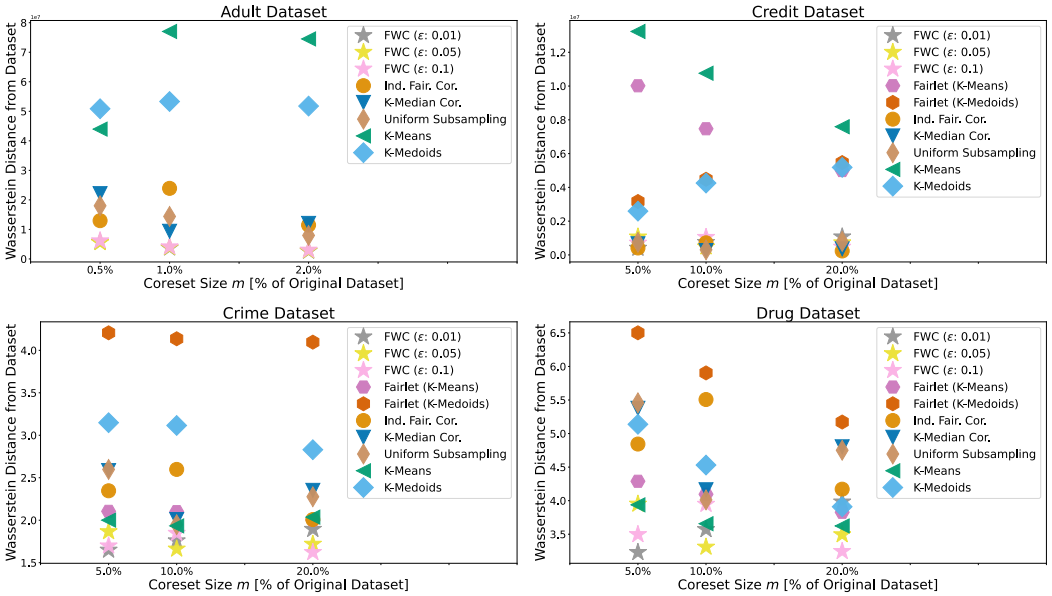

Figure 3: Wasserstein distance between the generated coreset and the original dataset (the lower values the better) for all methods, across the four real dataset. We report the averages over 10 separate train/test splits.

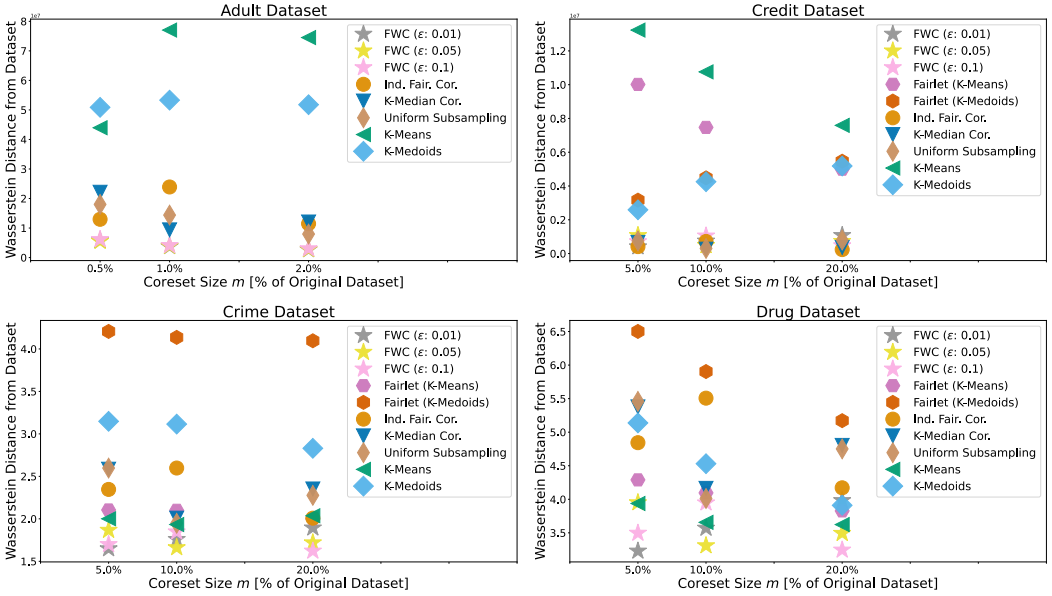

Figure 4: Clustering cost between the generated coreset and the original dataset (the lower values the better) for all methods, across the four real dataset. We report the averages over 10 separate train/test splits.

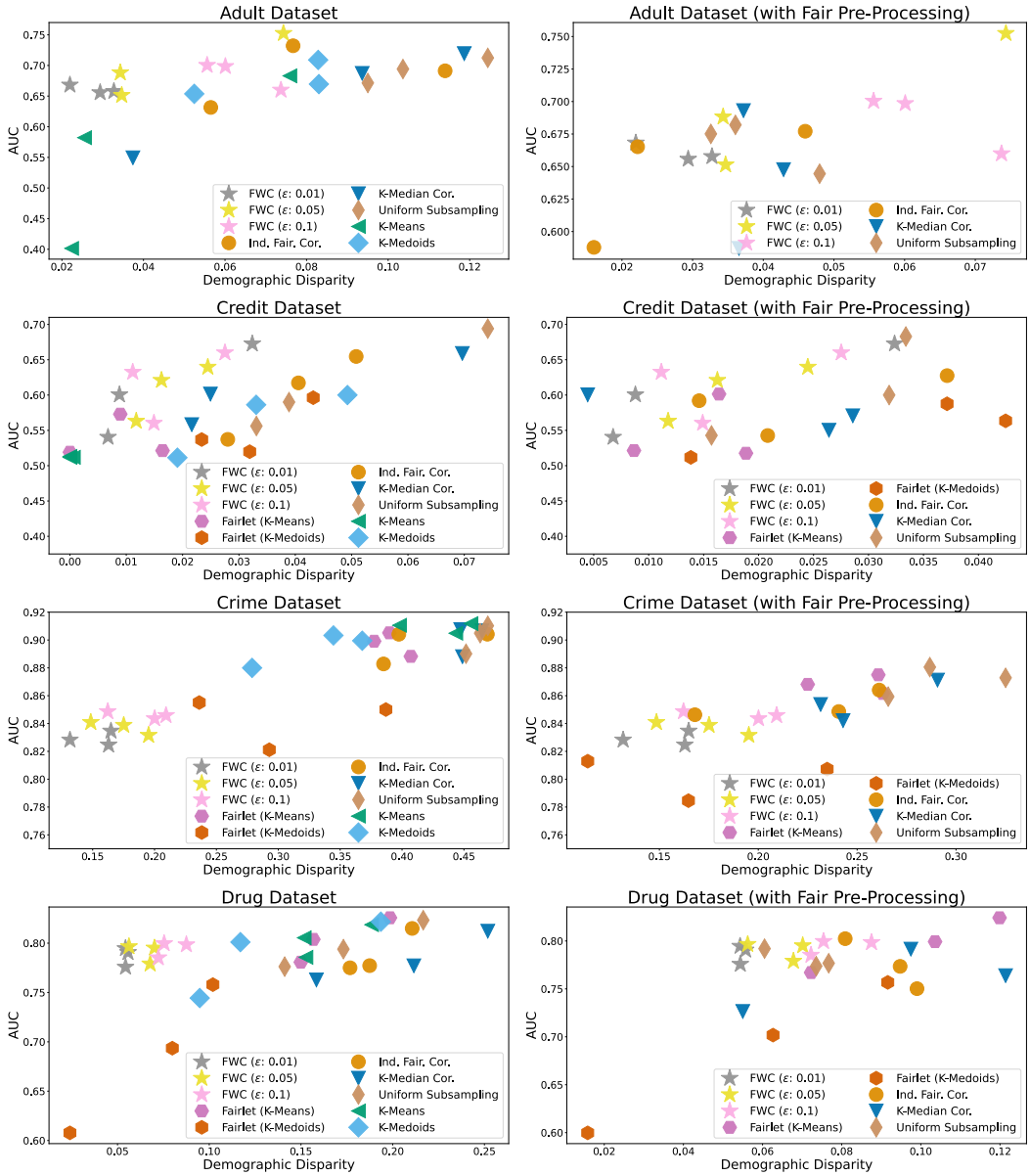

Figure 5: Fairness-utility tradeoff of all methods, indicated by AUC and demographic disparity of a downstream MLP classifier across all datasets (rows) and without (left column) or with (right column) fair pre-processing [21] (for all approaches apart from FWC). We report the averages over 10 separate train/test splits.

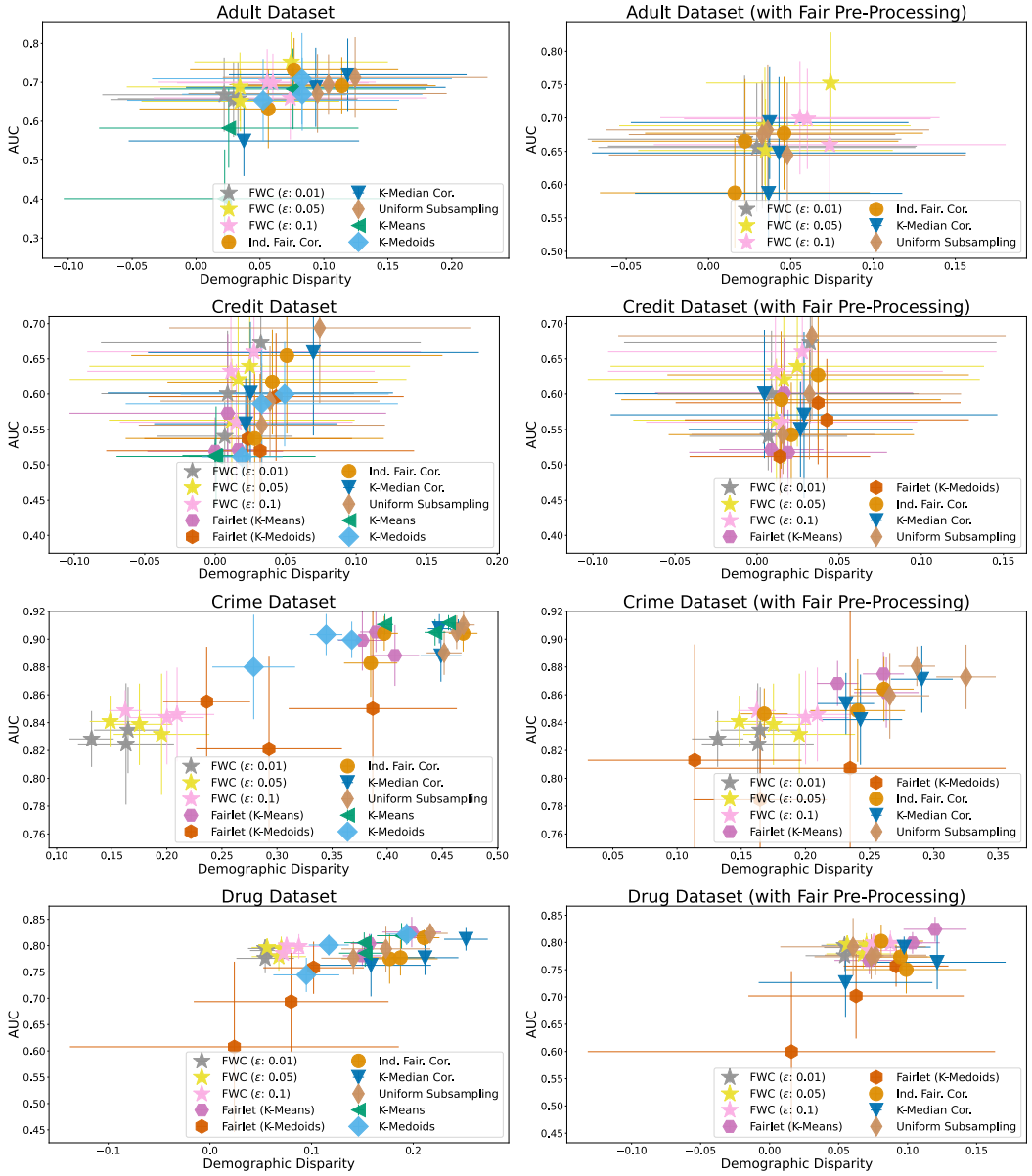

Figure 6: Same as Figure 5, but reporting averages and one standard deviations in both fairness and accuracy over 10 separate train/test splits.

	Wasserstein Distance				Clustering Cost			
Method	*Adult*	*Credit*	*Crime*	*Drug*	*Adult*	*Credit*	*Crime*	*Drug*
FWC (ϵ: 0.01)	2/**1**/**1**	**1**/5/6	**1**/2/3	**1**/2/6	6/5/5	**1**/7/9	**1**/4/7	**1**/6/7
FWC (ϵ: 0.05)	**1**/2/2	6/3/4	3/**1**/2	4/**1**/2	5/6/7	8/5/6	4/**1**/4	4/**1**/6
FWC (ϵ: 0.1)	3/3/3	4/6/3	2/3/**1**	2/4/**1**	7/7/6	6/8/4	2/7/2	2/8/2
Fairlet (K-Means)	-/-/-	9/9/7	5/7/4	5/6/4	-/-/-	3/2/2	5/3/3	5/5/3
Fairlet (K-Medoids)	-/-/-	8/8/9	10/10/10	10/10/10	-/-/-	10/10/10	10/10/10	10/10/9
Ind. Fair. Cor.	4/6/5	2/4/**1**	6/8/5	6/9/7	2/4/2	4/6/3	6/9/6	6/9/5
K-Median Cor.	6/4/6	3/**1**/2	7/6/8	8/7/9	4/2/3	5/3/5	9/6/9	9/4/10
Uniform Subsampling	5/5/4	5/2/5	8/5/7	9/5/8	3/3/4	7/4/7	8/5/8	8/2/8
K-Means	7/8/8	10/10/10	4/4/6	3/3/3	**1**/**1**/**1**	2/**1**/**1**	3/2/**1**	3/3/**1**
K-Medoids	8/7/7	7/7/8	9/9/9	7/8/5	8/8/8	9/9/8	7/8/5	7/7/4

Table 2: Ranking of coresets created by each methods in terms of Wasserstein distance to the original datasets and clustering cost, for each of the three coreset sizes m. `FWC` consistently creates coresets that are closer in distribution to the original dataset. For numeric values, see Figures 3 and 4.