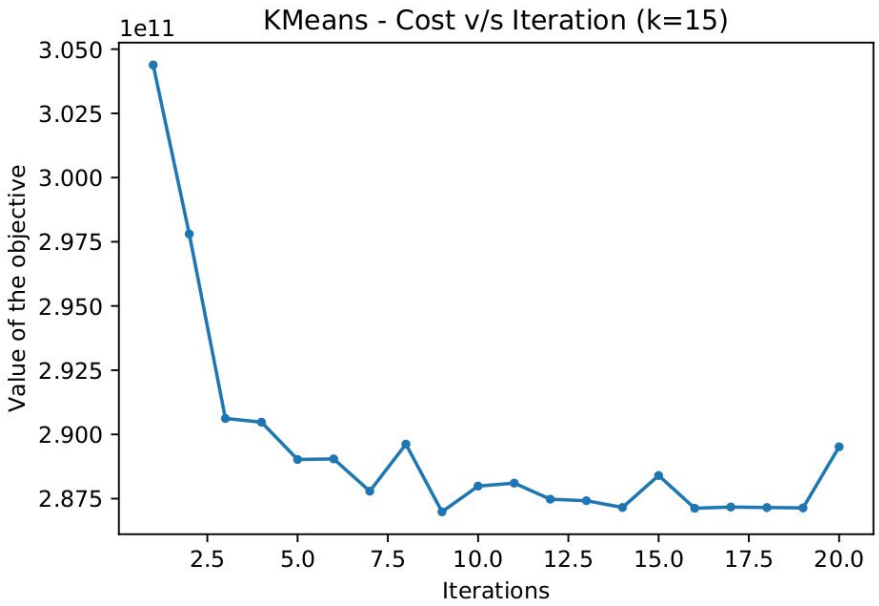
Data Science Project

Guided By :
Prof. Anirban Dasgupta

Group Members :
Apoorv Agnihotri
S Deepak Narayanan
Shivji Bhagat
Smeet Vora

Problem Statement

- To study efficient ways of sampling points from the dataset so as to speed up the clustering algorithms.



Dataset

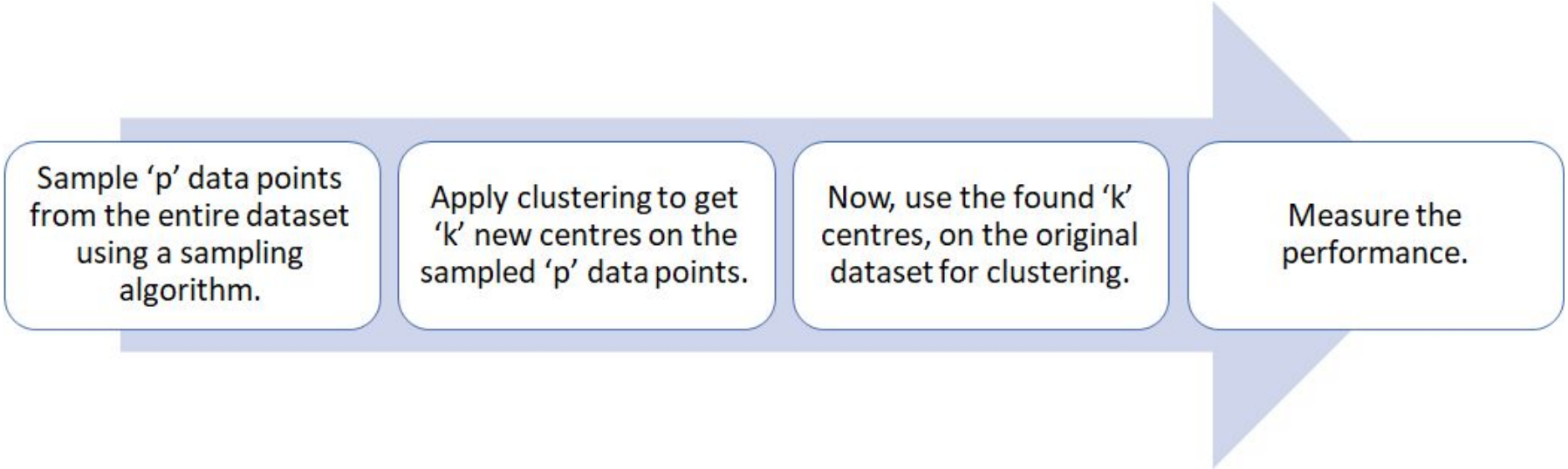
- **KDD Cup 2004:**
  Protein homology dataset

  <u>Specifications :</u>
  It measures the match between a protein and a native sequence.

  Total No. of Samples : ~1,45,000

  Number of features/dimensions : 74

Procedure


Sample 'p' data points from the entire dataset using a sampling algorithm.
Apply clustering to get 'k' new centres on the sampled 'p' data points.
Now, use the found 'k' centres, on the original dataset for clustering.
Measure the performance.

Approaches

Approaches : Uniform Random Sampling

- Baseline Sampling method
- Involves sampling ‘p’ data points from the entire data set uniformly at random with equal probabilities by varying the random seeds.

Approaches : Lightweight Coresets

Implemented the method suggested by Bachem et al. in “Scalable k-Means Clustering via Lightweight Coresets”, KDD 2018.

- Coresets
  - Weighted subsets of the dataset
  - Trained model competitive to the model trained on entire data
- Lightweight Coresets
  - Variation of Coresets
  - Admits both additive & multiplicative error
- Complexity - O (nd)

Algorithm 1 Lightweight coreset construction

Require: Set of data points $\mathcal{X}$, coreset size m

$\mu \leftarrow$ mean of $\mathcal{X}$

for $x \in \mathcal{X}$ **do**

$\quad q(x) \leftarrow \frac{1}{2}\frac{1}{|\mathcal{X}|} + \frac{1}{2}\frac{\mathrm{d}(x,\mu)^2}{\sum_{x' \in \mathcal{X}} \mathrm{d}(x',\mu)^2}$

end for

$C \leftarrow$ sample m weighted points from $\mathcal{X}$ where each point x has weight $\frac{1}{m \cdot q(x)}$ and is sampled with probability $q(x)$

Return lightweight coreset C

Approaches : Leverage Score Based Sampling

Leverage Score Based Sampling

1. In Leverage Score based sampling, we define probability distributions on the data points in our data matrix by using the QR Decomposition of the matrix.
2. We consider the QR Decomposition of the data matrix and define a distribution over the rows of the matrix. And, depending on the distribution on the Q matrix, we sample from the corresponding row of the original data matrix.

Approaches : Volume Sampling

Reverse Iterative Volume Sampling

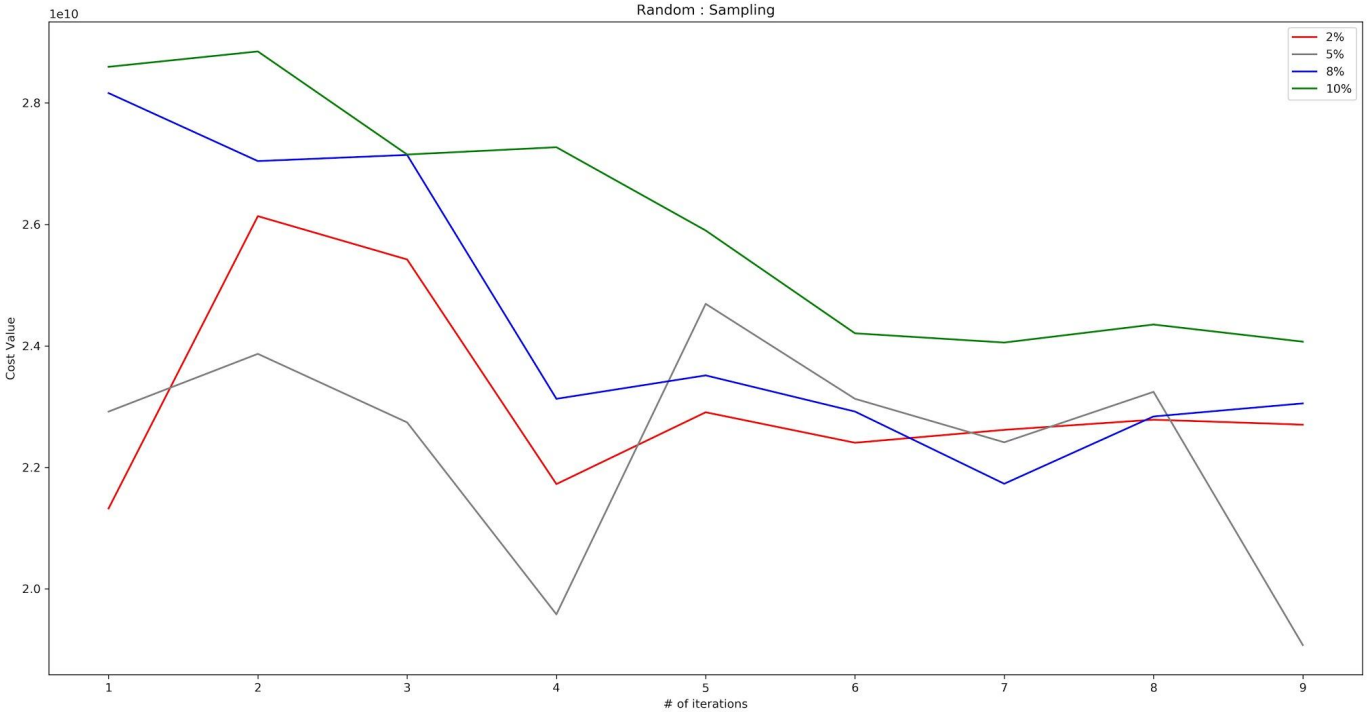
1. Implemented Derezinski et al., work "Unbiased estimates for linear regression via volume sampling", *NeurIPS 2017*.
2. Probability of sampling a subset of rows is proportional to the value of the determinant of (XX'), where X is the data matrix consisting of the subset of the rows of X.

Reverse iterative volume sampling

Input: $\mathbf{X} \in \mathbb{R}^{d \times n}$, $s \in \{d..n\}$
$\mathbf{Z} \leftarrow (\mathbf{X}\mathbf{X}^\top)^{-1}$
$\forall_{i \in \{1..n\}} \quad p_i \leftarrow 1 - \mathbf{x}_i^\top \mathbf{Z} \mathbf{x}_i$
$S \leftarrow \{1, .., n\}$
while $|S| > s$
  Sample $i \propto p_i$ out of S
  $S \leftarrow S - \{i\}$
  $\mathbf{v} \leftarrow \mathbf{Z}\mathbf{x}_i / \sqrt{p_i}$
  $\forall_{j \in S} \quad p_j \leftarrow p_j - (\mathbf{x}_j^\top \mathbf{v})^2$
  $\mathbf{Z} \leftarrow \mathbf{Z} + \mathbf{v}\mathbf{v}^\top$
end
return S

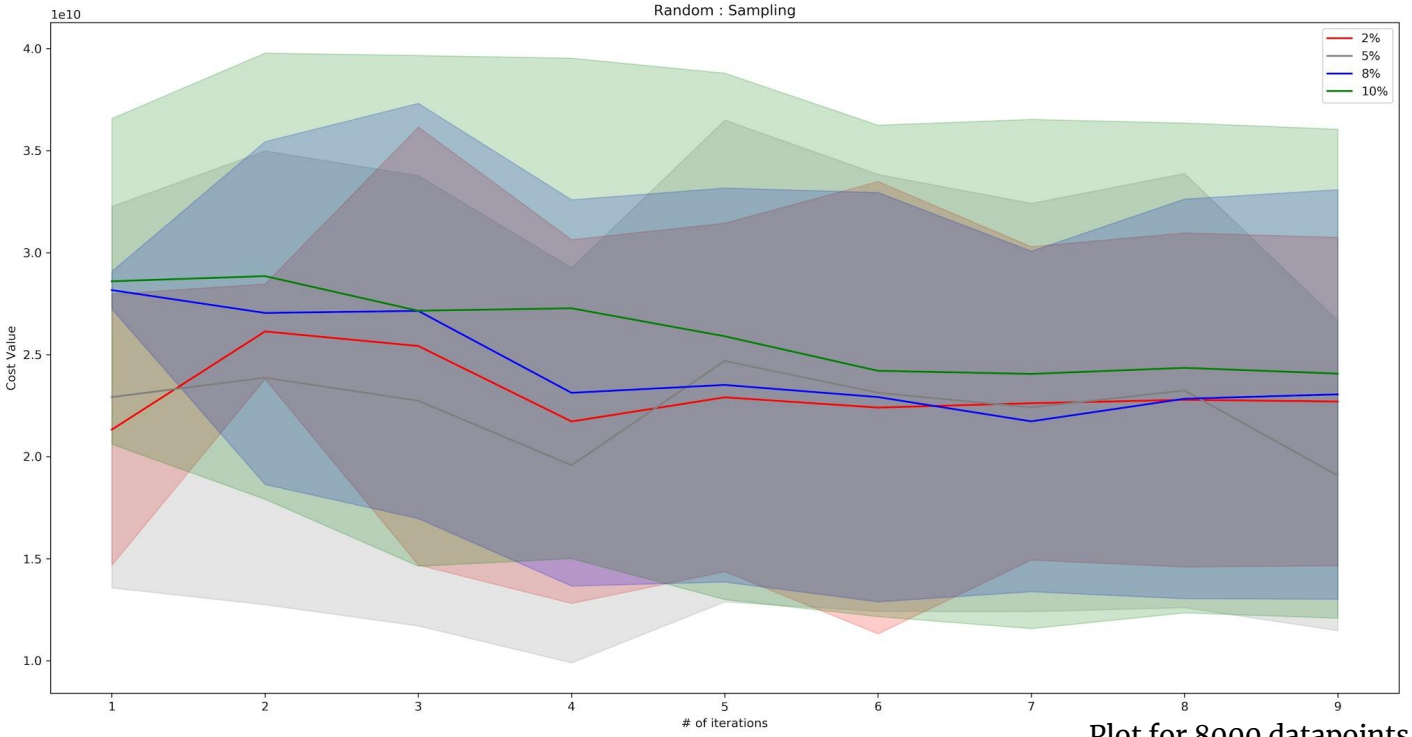
Results

Results : Random Sampling



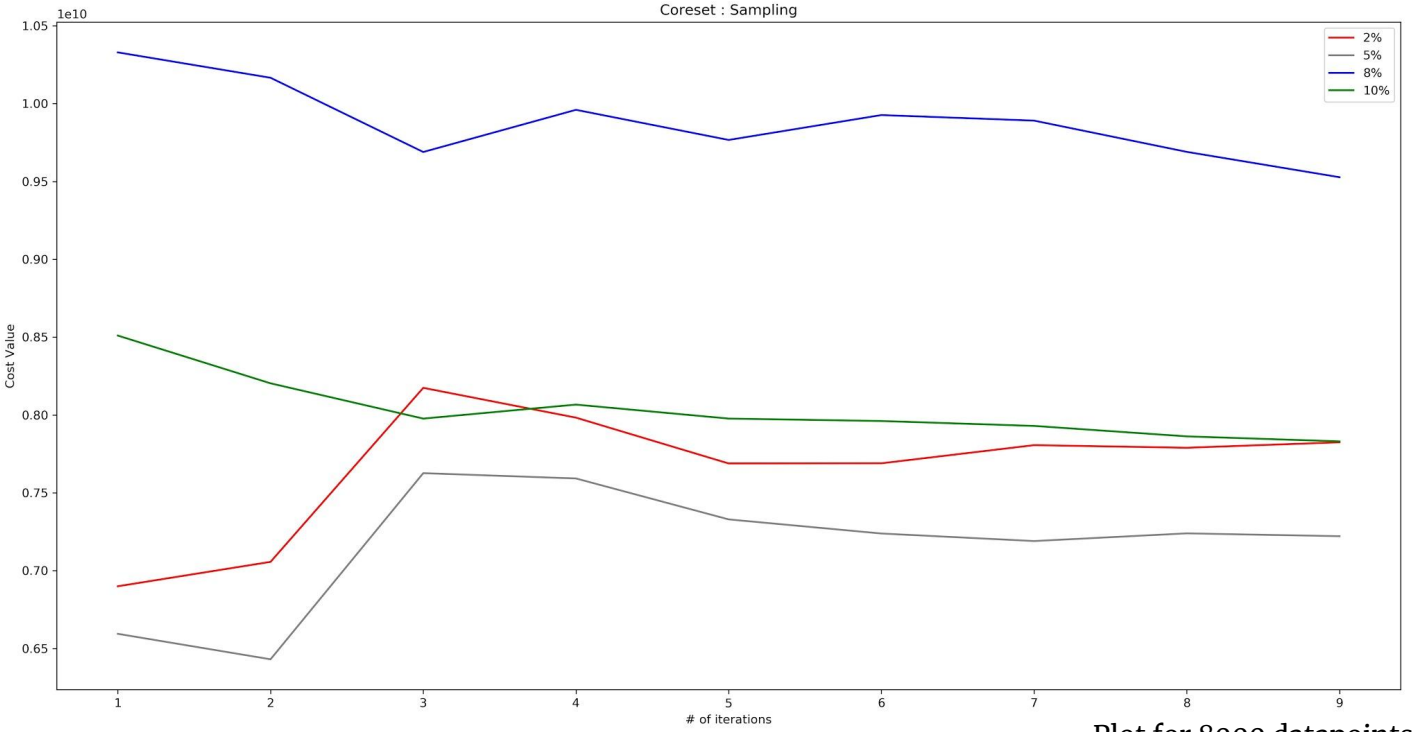
Plot for 8000 datapoints

Results : Random Sampling with std dev



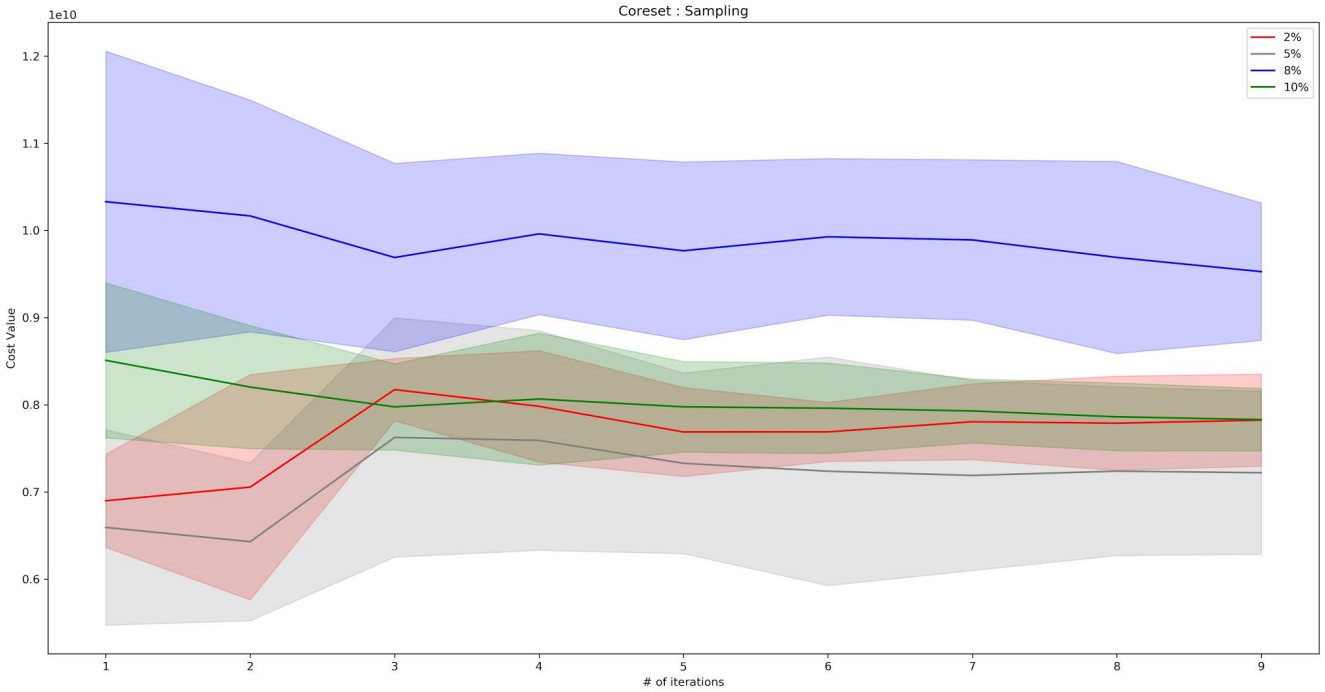
Plot for 8000 datapoints

Results : Lightweight Coresets



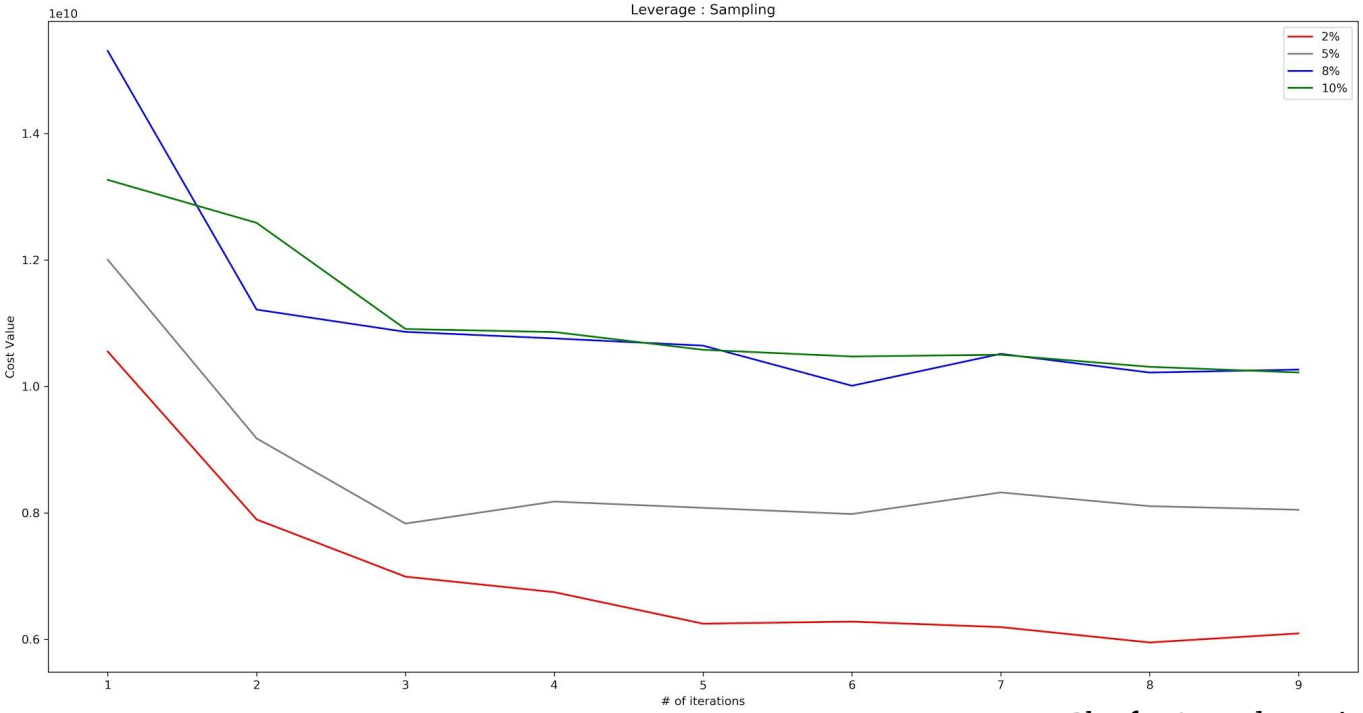
Plot for 8000 datapoints

Results : Lightweight Coresets with std dev



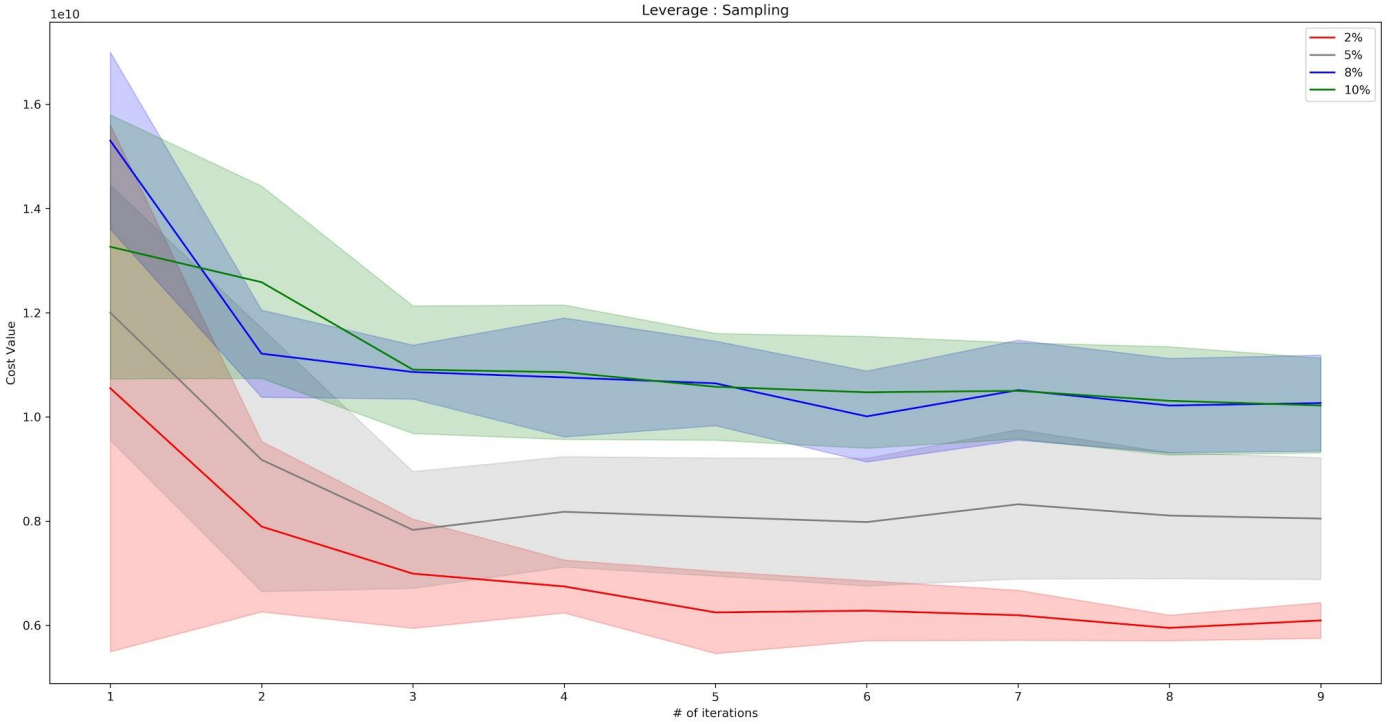
Plot for 8000 datapoints

Results : Leverage Sampling



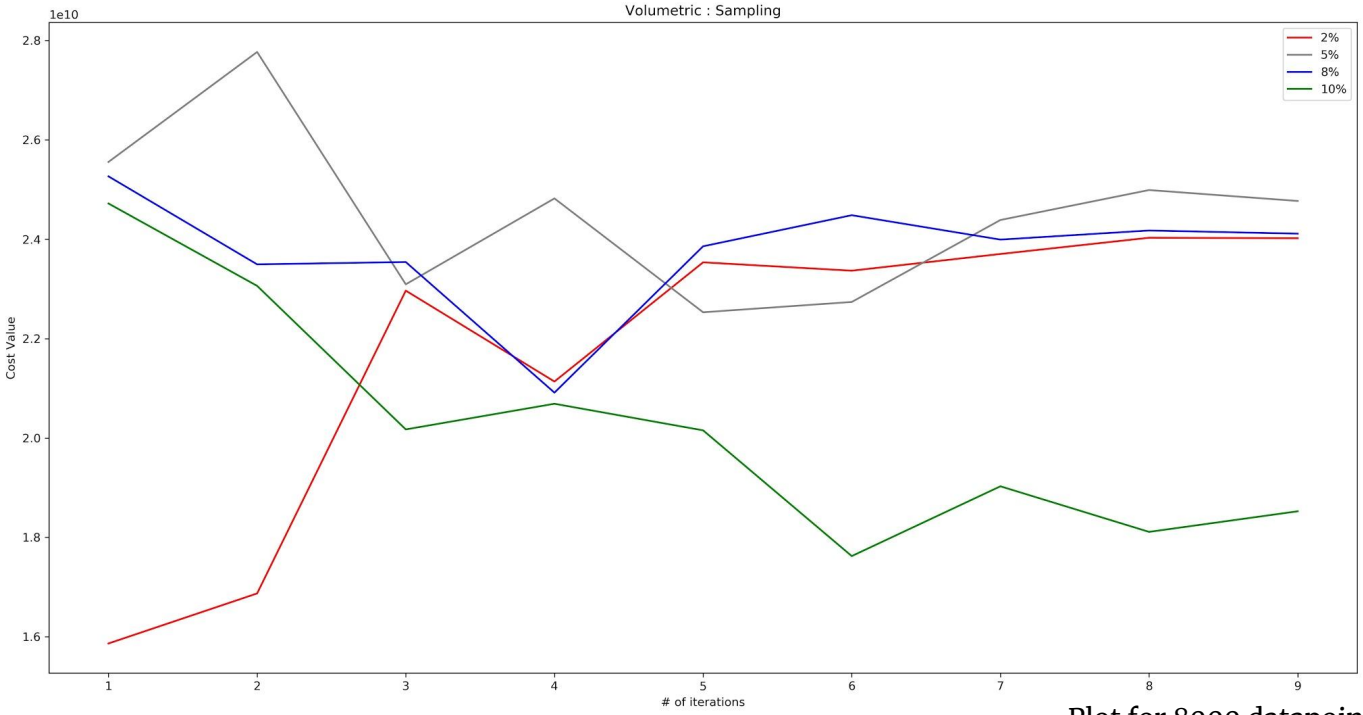
Plot for 8000 datapoints

Results : Leverage Sampling with std dev



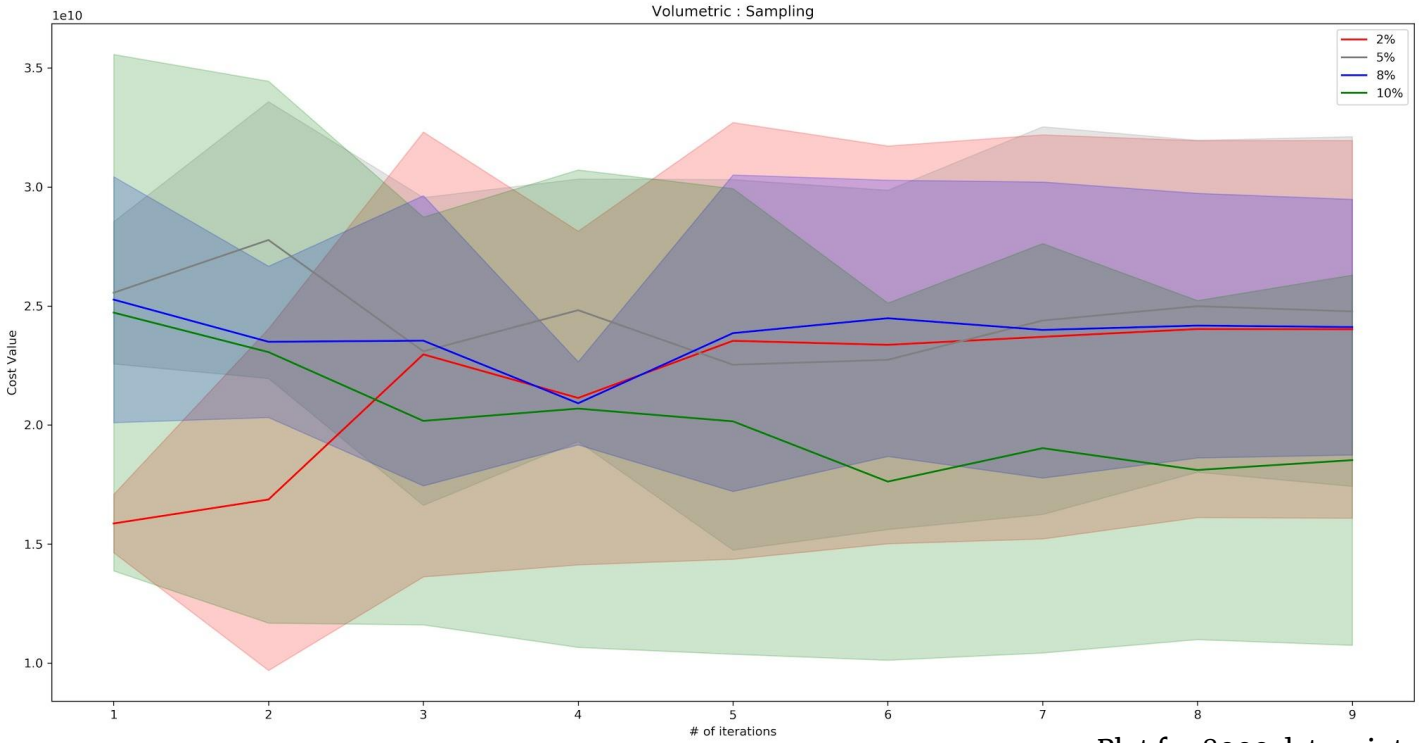
Plot for 8000 datapoints

Results : Volumetric Sampling



Plot for 8000 datapoints

Results : Volumetric Sampling with std dev

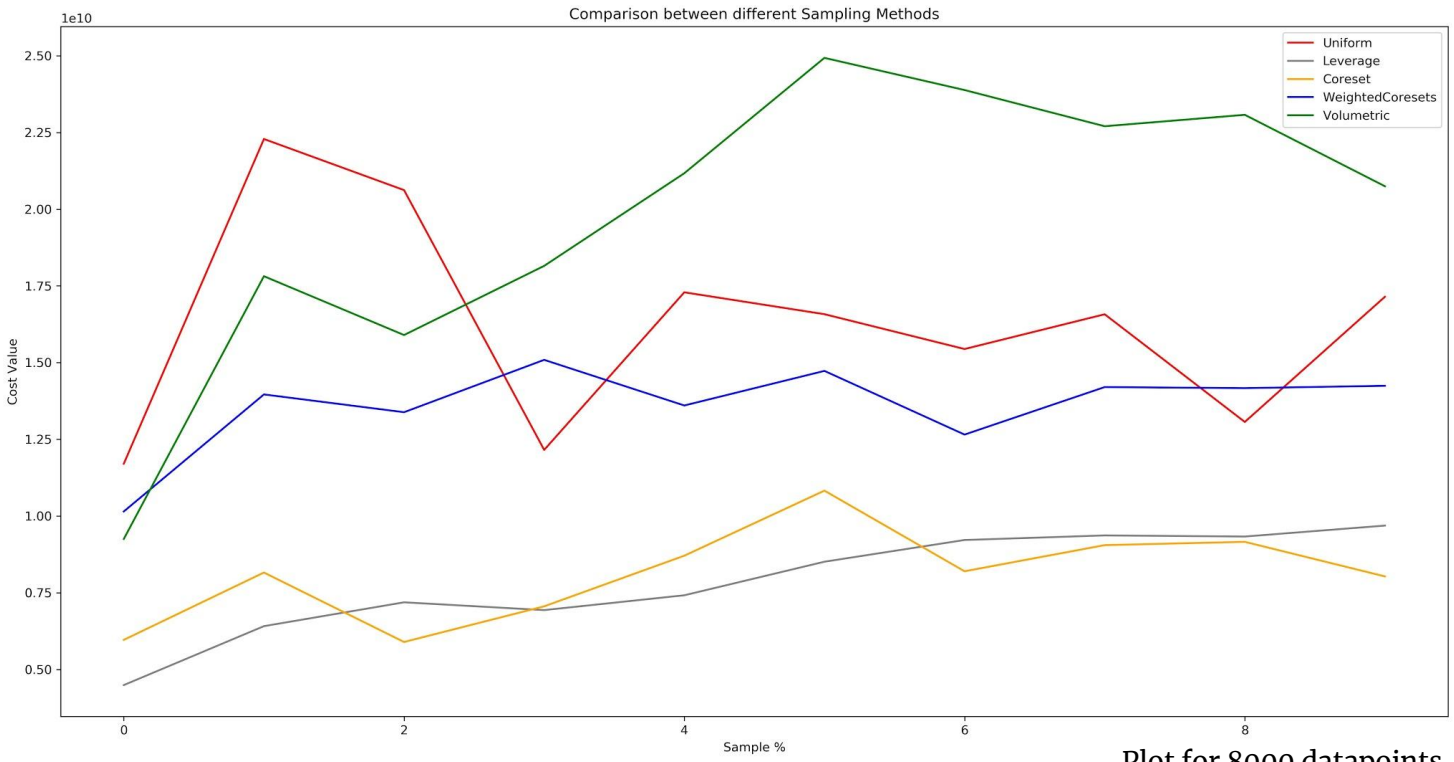


Plot for 8000 datapoints

Results on Entire Dataset !

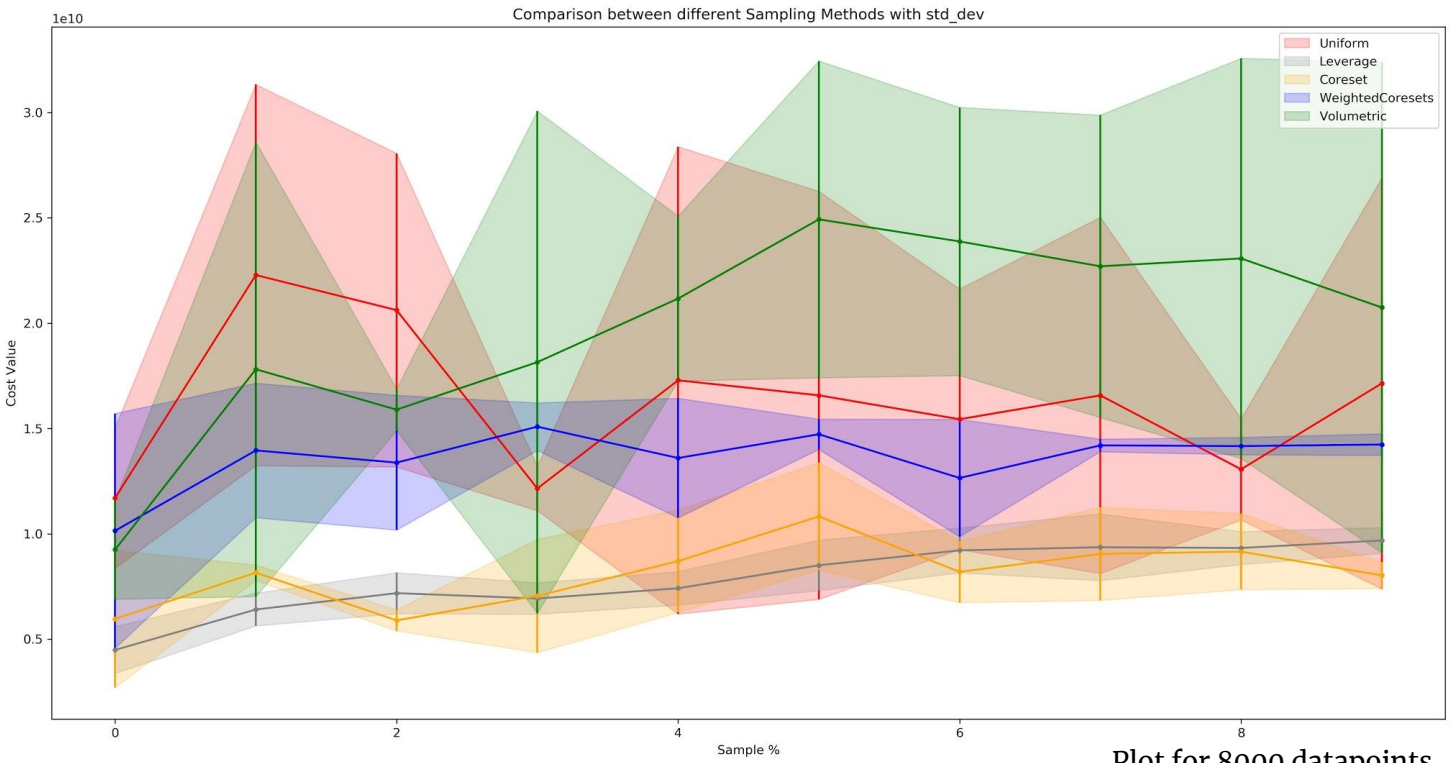
```
apoorv-agnihotri@tensorflow-1-vm: ~/cpp/Data_Science_Project
File  Edit  View  Search  Terminal  Tabs  Help
apoorv@apoo-ubu18: ~/Desktop/github/Data_Science_Project ×   apoorv-agnihotri@tensorflow-1-vm: ~/cpp/Data_Science_Pro... ×

  1  [|                3.2%]   9  [                0.0%]   17 [                0.0%]   25 [                0.0%]
  2  [||||||||||||100.0%]   10 [                0.0%]   18 [                0.0%]   26 [                0.0%]
  3  [                0.0%]   11 [                0.0%]   19 [                0.0%]   27 [                0.0%]
  4  [                0.0%]   12 [||||||||||||100.0%]   20 [                0.0%]   28 [                0.0%]
  5  [                0.0%]   13 [                0.0%]   21 [                0.0%]   29 [                0.0%]
  6  [                0.0%]   14 [                0.0%]   22 [                0.0%]   30 [||||||||||||100.0%]
  7  [                0.0%]   15 [                0.0%]   23 [                0.0%]   31 [||               1.3%]
  8  [||||||||||||100.0%]   16 [                0.0%]   24 [                0.0%]   32 [                0.0%]
  Mem[|||||                                3.09G/118G]   Tasks: 106, 213 thr; 5 running
  Swp[                                          0K/0K]   Load average: 4.38 4.20 4.11
                                                         Uptime: 13:43:54

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
17025 apoorv-ag  21   1 1940M  289M 12248 R 99.7  0.2  9h18:43 python3 Final_Sampling-Individual.py
12433 apoorv-ag  23   3 2262M  352M 13160 R 99.7  0.3 12h33:51 python3 Final_Sampling.py
17026 apoorv-ag  22   2 2177M  263M 13260 R 99.7  0.2  9h21:58 python3 Final_Sampling-Individual.py
12432 apoorv-ag  24   4 2100M  562M  6516 R 99.0  0.5 12h33:39 python3 Final_Sampling.py
18683 apoorv-ag  20   0 25092  4432  3112 R  0.7  0.0  0:00.20 htop
 1002 root       20   0 2569M 36144 24316 S  0.7  0.0  0:28.97 /usr/bin/containerd
 1005 root       20   0 2977M 64272 37876 S  0.0  0.1  0:40.02 /usr/bin/dockerd -H fd:// --containerd=/run/c
 2492 root       20   0 2977M 64272 37876 S  0.0  0.1  0:01.12 /usr/bin/dockerd -H fd:// --containerd=/run/c
15409 root       20   0 2569M 36144 24316 S  0.0  0.0  0:00.50 /usr/bin/containerd
    1 root       20   0 57064  6812  5252 S  0.0  0.0  0:02.90 /sbin/init
  679 root       20   0 46092  8972  8456 S  0.0  0.0  0:00.54 /lib/systemd/systemd-journald
  714 root       20   0 45652  3760  2856 S  0.0  0.0  0:00.23 /lib/systemd/systemd-udevd
  803 root       20   0  4204   708   648 S  0.0  0.0  0:00.00 /usr/sbin/acpid
  808 messagebu  20   0 45120  3724  3284 S  0.0  0.0  0:00.13 /usr/bin/dbus-daemon --system --address=syste
  817 root       20   0 29636  2752  2504 S  0.0  0.0  0:00.07 /usr/sbin/cron -f
  830 root       20   0 46496  4660  4104 S  0.0  0.0  0:00.23 /lib/systemd/systemd-logind
  947 root       20   0 20472  2968  1876 S  0.0  0.0  0:00.00 /sbin/dhclient -4 -v -pf /run/dhclient.eth0.p
 1022 root       20   0 2569M 36144 24316 S  0.0  0.0  0:05.13 /usr/bin/containerd
 1023 root       20   0 2569M 36144 24316 S  0.0  0.0  0:00.04 /usr/bin/containerd
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Comparison between different sampling methods



Plot for 8000 datapoints

Comparison between diff. sampling methods with std dev



Plot for 8000 datapoints

References

1. Olivier Bachem, Mario Lucic, Andreas Krause. Scalable k-Means Clustering via Lightweight Coresets, In KDD 2018
2. KDD Cup 2004, Protein Homology Dataset.
3. M. Derezinski et al, Unbiased estimates for linear regression via volume sampling, *In NeurIPS 2017*
4. Amit Deshpande, Luis Rademacher, Santosh Vempala, and Grant Wang. Matrix approximation and projective clustering via volume sampling. In Proceedings of the Seventeenth Annual ACM-SIAM Symposium on Discrete Algorithm, SODA ’06.