A Implementation details

A.1 Proposed algorithm

The key idea of this algorithm is to compute the proposed value function Equation 3 for each given class c (line 4). We implemented the approximation to the canonical CS-SHAPLEY suggested in Equation 6 and in practice we found that $K = 500$ is sufficient for our experiments. Lines 10 – 18 is adopted from the truncated Monte Carlo algorithm [3] for computing the Shapley values for the training instances in class c, with $\varepsilon = 10^{-4}$. Note that, the approximation to the canonical CS-SHAPLEY also requires drawing different subsets $S_{-c}^{(k)} \in T_{-c}$ for each k (line 7), which, based on our observation, is critical for estimating reliable CS-SHAPLEY values.

Algorithm 1: CS-SHAPLEY

1: **Input:** Train data $T = \{1, \ldots, n\}$, label set C, development set D
2: **Output:** Shapley value of training points: $\phi_1, \ldots, \phi_n$
3: Initialize $\phi_i \leftarrow 0$ for $i \in \{1, \ldots, n\}$
4: **for** $c \in C$ **do**
5: **for** $k \in \{1, \ldots, K\}$ **do**
6: $\pi_c^{(k)}$: A random permutation of the training instances with label c, T_c
7: $S_{-c}^{(k)} \subseteq T_{-c}$: A random subset of training instances that in other classes
8: Set $v_c(\emptyset | S_{-c}^{(k)}) \leftarrow 0$
9: Compute $v_c(T_c | S_{-c}^{(k)})$ using Equation 3
10: **for** $j \in \{1, \ldots, |T_c|\}$ **do**
11: *// $\pi_c^{(k)}[1 : j-1]$ represents the first $j-1$ examples in $\pi_c^{(k)}$*
12: **if** $v_c(T_c | S_{-c}^{(k)}) - v_c(\pi_c^{(k)}[1 : j-1] | S_{-c}^{(k)})| < \varepsilon$ **then**
13: $v_c(\pi_c^{(k)}[1 : j] | S_{-c}^{(k)}) \leftarrow v_c(\pi_c^{(k)}[1 : j-1] | S_{-c}^{(k)})$
14: **else**
15: Compute $v_c(\pi_c^{(k)}[1 : j] | S_{-c}^{(k)})$ using Equation 3
16: **end if**
17: $\phi_{\pi_c^{(k)}[j]} \leftarrow \frac{k-1}{k} \phi_{\pi_c^{(k-1)}[j]} + \frac{1}{k} \{ v_c(\pi_c^{(k)}[1 : j] | S_{-c}^{(k)}) - v_c(\pi_c^{(k)}[: j-1] | S_{-c}^{(k)}) \}$
18: **end for**
19: **end for**
20: *// Normalize the values of the instances in class c to satisfy the efficiency axiom*
21: $\sigma_c \leftarrow \sum_{j \in T_c} \phi_j$
22: $\phi_j \leftarrow \frac{\phi_j}{\sigma_c} \cdot a_T(D_c)$ for $j \in T_C$
23: **end for**

A.2 Data usage

We use nine real datasets, including five tabular datasets and four image datasets: Diabetes, Click, CPU, Covertype, Phoneme, FMNIST, CIFAR10, and MNIST (binarized and multi-class). All datsets are licensed under CC-BY. Dataset statistics and sources are provided in Table 3. Datasets do not contain offensive or personally identifiable information. For any consent information, if available, please see the cited data sources.

For each dataset, we maintain the original label distribution by performing stratified sampling. For all datasets other than Diabetes, we sample a 500/500/2000 train/dev/test split. Since the Diabetes dataset only contains 768 total instances, we use a 128/128/512 train/dev/test split. Split ratios are based on common ratios from prior works [14, 3].

For MNIST [16], CIFAR-10 [13], and Fashion MNIST [25] we follow the common procedure of prior work [14, 15]: we extract the learned image representations from ResNet-18 [8] using the pre-trained weights available from Pytorch [19]. Then, we fit a principal component analysis model on the extracted feature vectors and select the first 32 principal components. As prior work has

Table 3: Statistics for each dataset. Input dimensions refers to dimensionality of input (x) without class label (y).

Dataset	Data Type	Classes	Input Dims	Source
Diabetes	Tabular	2	8	`https://www.openml.org/d/37`
Click	Tabular	2	11	`https://www.openml.org/d/1216`
CPU	Tabular	2	21	`https://www.openml.org/d/197`
Covertype	Tabular	7	54	`https://www.openml.org/d/1596`
Phoneme	Tabular	2	5	`https://www.openml.org/d/1489`
FMNIST	Image	2	32	`https://www.openml.org/d/40996`
CIFAR10	Image	2	32	`https://www.openml.org/d/40927`
MNIST (binary)	Image	2	32	`https://www.openml.org/d/554`
MNIST (multi)	Image	10	32	`https://www.openml.org/d/554`

used binarized image datsets [14, 3], we binarize the datasets as follows: $y = \{\text{automobile}, \text{truck}\}$ (CIFAR10), $y = \{\text{t-shirt and tops}, \text{shirts}\}$ (FMNIST), $y = \{1, 7\}$ (MNIST).

A.3 Experiment settings

We use the model implementations from the Python module `scikit-learn` [5], licensed under BSD 3-Clause License. Experiments primarily use `scikit-learn` default hyperparameters with some preliminary hyperparameter tuning. For example, as we use small datasets, we perform preliminary hyperparameter tuning of the logistic regression solver between the default 'lbfgs' and 'liblinear', selecting 'liblinear' as it had higher validation set performance across datasets. For the gradient boosting classifier, we reduced the size and number of estimators to maintain computational feasibility for all methods (see section 5 for example of computational scaling), using 40 estimators with a maximum depth of 2 and 6 required samples to split internal nodes. Experiments were run on 65 Intel(R) Xeon(R) Silver 4110 CPUs 2.10GHz.

For all reported results we use a development set to compute the train data values, then evaluate model performance on a held-out test set. Further, results represent the mean of 5 trials. In plots, the shaded region indicates standard deviation.

[5] `https://scikit-learn.org/stable/`

B Theoretical justification

B.1 Additional discussion

Development set level metrics: In addition to the issues with development set level accuracy, using another development set level metric (e.g. $F1$) does not resolve the issue of being able to distinguish between in-class and out-of-class contribution. In other words, the same issues are present with other performance metrics as with accuracy in the original utility function. We offer an example in Table 4 and Table 5:

Table 4: Confusion matrix prior to adding a noisy instance from the positive class.

		Predicted Label	
		+	-
Ground Truth	+	3	2
	-	2	3

Table 5: Confusion matrix after adding a noisy instance from the positive class.

		Predicted Label	
		+	-
Ground Truth	+	2	3
	-	1	4

In Table 4, prior to adding a noisy data point, we have precision$= 0.6$, recall$= 0.6$, so $F1 = 0.6$. In Table 5, we have precision$= 0.7$, recall$= 0.6$, so $F1 = 0.65$. The contribution of this instance is positive, even though it decreased the prediction performance of its own class, which is the same conceptual issue encountered with development set level accuracy that is exemplified in Figure 1.

C Additional results

In the following subsections, we report the results across experiments for the KNN and gradient boosting classifiers. We also include the additional plots on removal and transferabilitiy for logistic regression and SVM with the RBF kernel.[6]

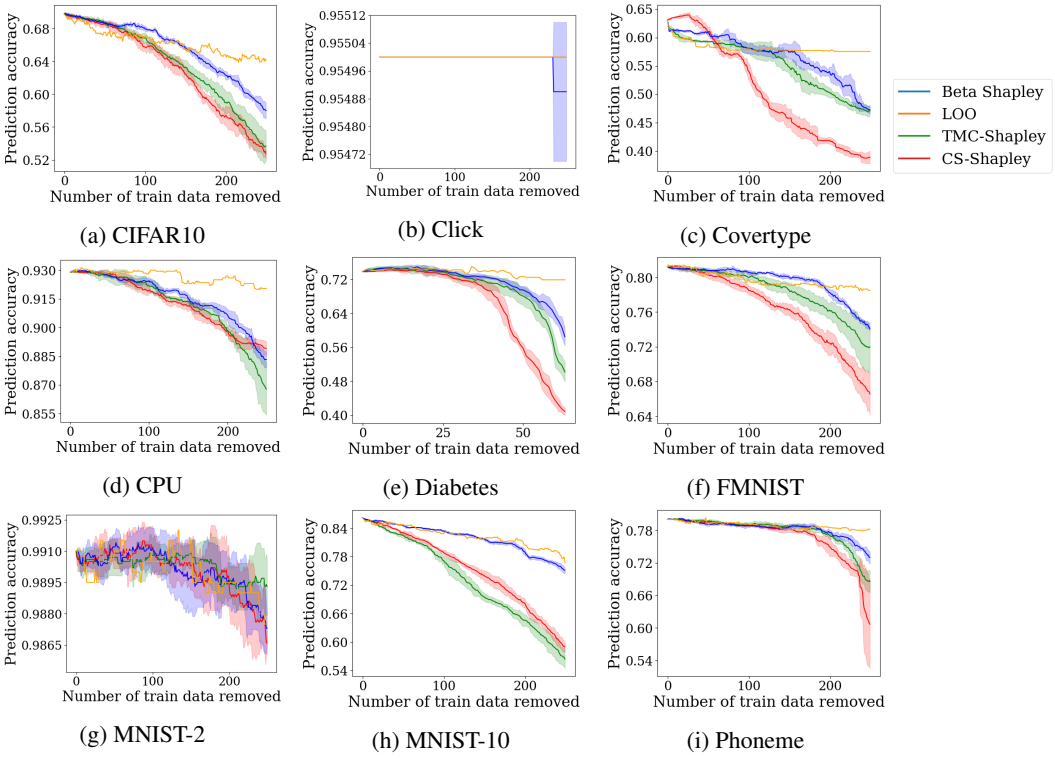
C.1 Additional removal results

We report the weighted accuracy drop (WAD) for gradient boosting classifier and KNN and plot the prediction accuracy as a function of training instances removed for SVM-RBF, gradient boosting classifier, and KNN.

Table 6: Weighted accuracy drop for gradient boosting classifier and KNN using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Gradient Boosting				KNN			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.097**	0.027	0.053	-0.027	**0.108**	0.084	0.090	0.024
Click	**0.029**	0.017	0.026	0.012	**0.050**	0.008	0.007	0.002
Covertype	**0.142**	0.101	0.029	-0.014	**0.232**	0.195	0.150	0.002
CPU	0.021	0.008	0.008	**0.046**	**0.053**	0.028	0.027	0.044
Diabetes	**0.117**	0.095	0.055	0.091	**0.226**	0.080	0.050	0.069
FMNIST	**0.117**	0.079	0.089	0.047	**0.079**	0.034	0.024	0.058
MNIST-2	**0.018**	0.006	0.012	0.012	0.010	**0.018**	0.012	0.013
MNIST-10	–	–	–	–	0.184	**0.241**	0.100	0.105
Phoneme	**0.073**	0.028	0.039	0.020	**0.064**	0.042	0.037	0.048

[6]Gradient boosting classifier on MNIST-10 is computationally prohibitive and therefore not reported.

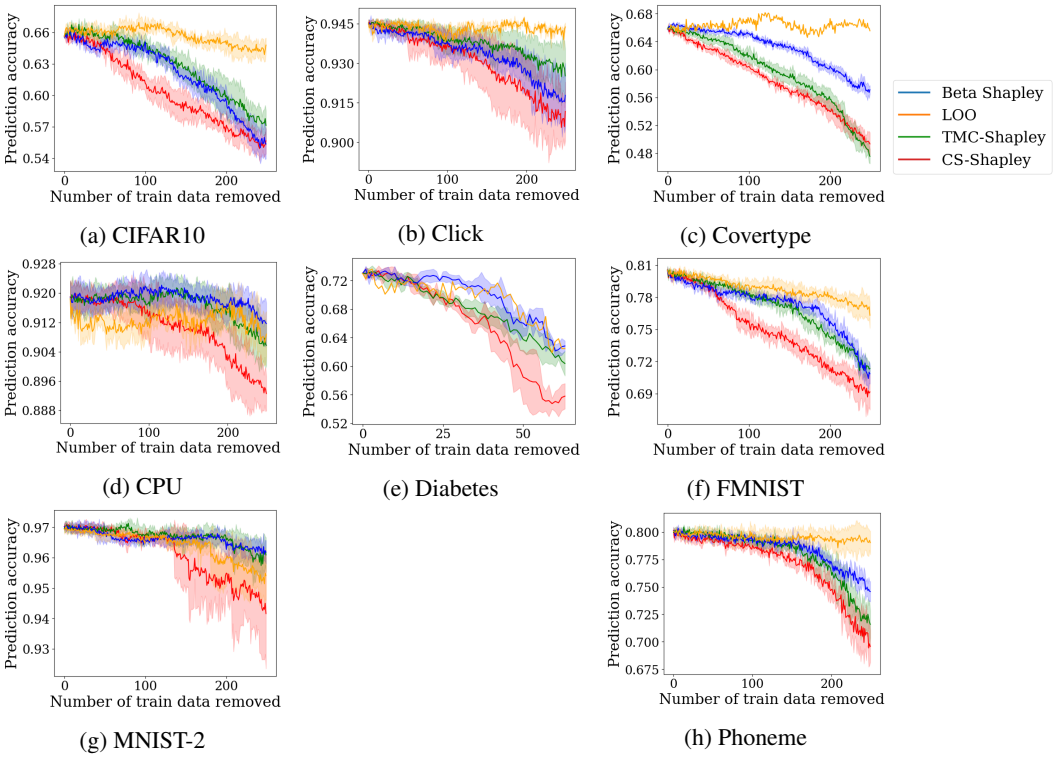


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 5: Performance across datasets when removing high-value instances for SVM with RBF Kernel.

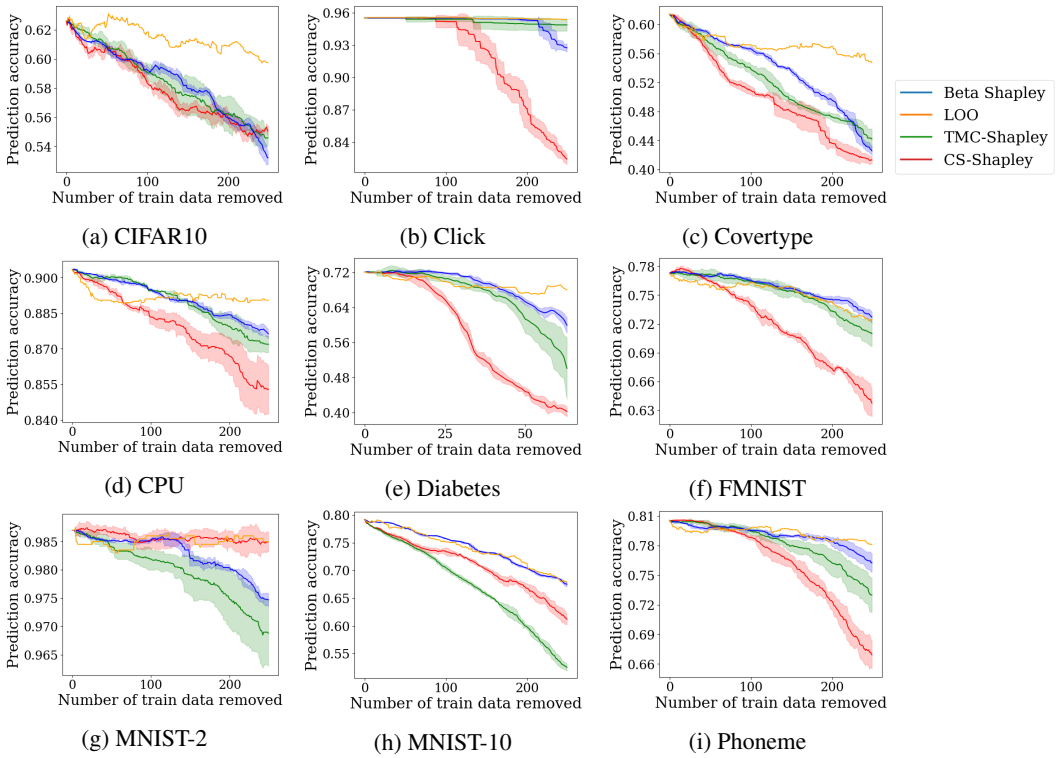


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) Phoneme

Figure 6: Performance across datasets when removing high-value instances for gradient boosting classifier.



(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

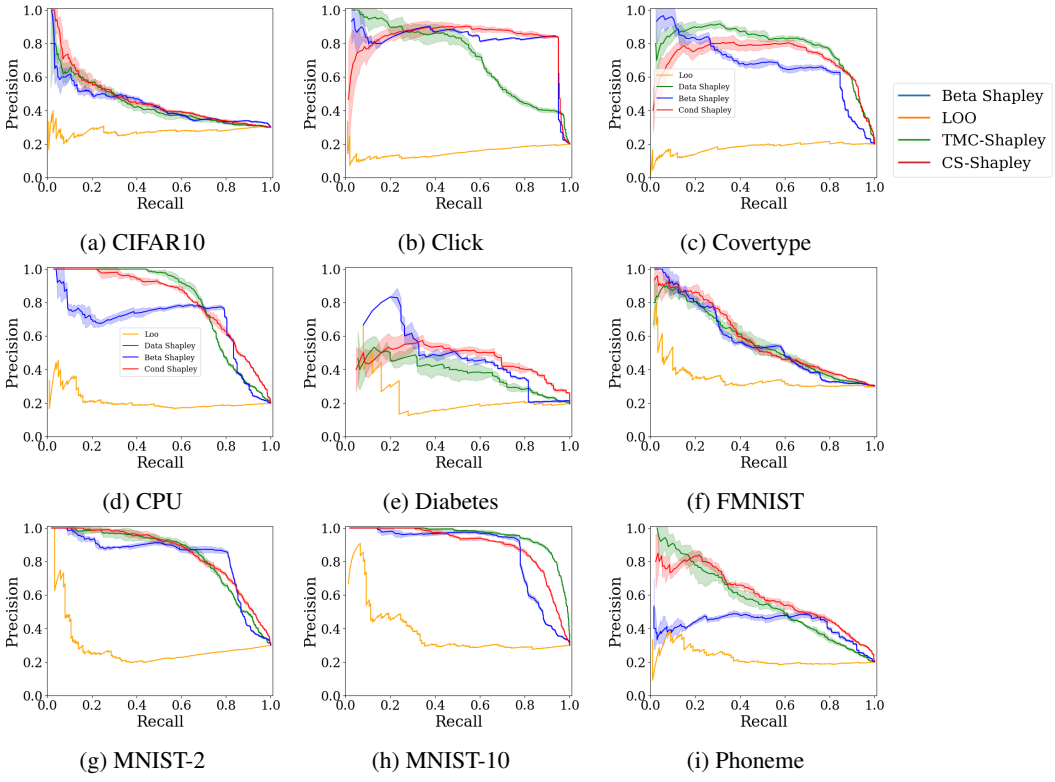
Figure 7: Performance across datasets when removing high-value instances for KNN.

C.2 Additional noisy label detection results

We report the AUC for gradient boosting classifier and KNN. To help visualize the relationship between AUC and the PR-curve, we also include the PR-curves for the logistic regression and SVM-RBF AUC results reported in the main content. Overall, relative performance ranking of the methods is consistent for this task and high-removal task for gradient boosting classifier and is variable on KNN.

Table 7: Area Under the Curve (AUC) for gradient boosting classifier and KNN using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Gradient Boosting				KNN			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.328**	0.318	0.311	0.297	0.324	0.313	**0.361**	0.299
Click	**0.774**	0.768	0.678	0.197	0.746	**0.808**	0.803	0.219
Covertype	0.627	**0.711**	0.598	0.182	**0.685**	0.440	0.437	0.232
CPU	**0.664**	0.624	0.456	0.193	0.558	**0.632**	0.447	0.187
Diabetes	**0.327**	0.319	0.288	0.198	**0.330**	0.320	0.320	0.192
FMNIST	**0.416**	0.377	0.374	0.300	0.364	0.453	**0.503**	0.319
MNIST-2	0.584	**0.604**	0.500	0.277	0.705	0.728	**0.806**	0.263
MNIST-10	–	–	–	–	0.796	**0.839**	0.764	0.326
Phoneme	**0.522**	0.510	0.414	0.197	0.457	**0.479**	0.429	0.180

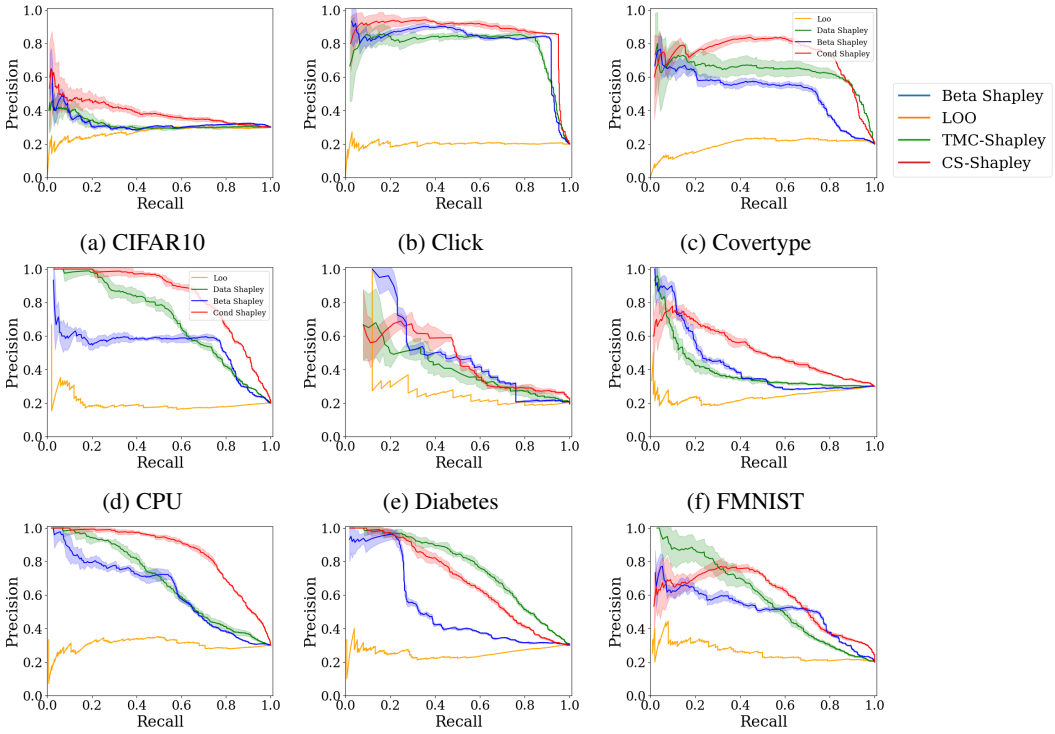


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 8: PR-Curve for noisy label detection using logistic regression.



(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 9: PR-Curve for noisy label detection using SVM with the RBF kernel.

C.3 Additional transferability of data values results

While we are primarily interested in transfer performance to otherwise computationally infeasible classifiers (i.e. transfer to MLP), we report all transfer combinations to gauge general patterns of transferability across even relatively simple classifiers. Specifically, we report the weighted accuracy drop WAD for each source classifier (logistic regression, SVM-RBF, gradient boosting classifier, and KNN) when transferring to each target classifier (logistic regression, SVM-RBF, gradient boosting classifier, KNN, MLP). Additionally, we show these results visually by plotting the prediction accuracy as a function of training instances removed. Note that results where the source classifier and target classifier are the same are reported in the high-value data removal task. Overall, KNN shows the weakest potential for value transferability. Performance is generally weaker (including more instances where the LOO simple baseline outperforms all Shapley-based methods), and the accuracy drop is generally lower and less stable from the source classifier across most target classifiers in comparison to other source classifiers.

Table 8: Weighted accuracy drop for transferring logistic regression values to SVM-RBF and KNN using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	SVM-RBF				KNN			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.094**	0.092	0.071	0.035	0.055	0.070	**0.079**	-0.016
Click	0.004	0.004	0.004	0.004	**0.039**	0.005	0.004	0.002
Covertype	**0.210**	0.187	0.115	0.073	**0.248**	0.187	0.109	0.126
CPU	**0.030**	0.014	0.013	0.022	**0.029**	0.012	0.008	-0.001
Diabetes	**0.113**	0.048	0.010	0.008	**0.182**	0.066	0.051	-0.008
FMNIST	**0.068**	0.060	0.025	0.033	0.015	**0.030**	0.015	0.005
MNIST-2	**0.103**	0.007	0.009	0.010	0.008	0.009	0.006	**0.010**
MNIST-10	0.186	**0.201**	0.080	0.076	0.140	**0.197**	0.083	0.052
Phoneme	**0.145**	0.023	0.021	0.041	**0.122**	0.022	0.019	0.024

Table 9: Weighted accuracy drop for transferring logistic regression values to gradient boosting classifier and multi-layer perceptron (MLP) using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Gradient Boosting				MLP			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.111**	0.082	0.045	0.001	**0.101**	0.096	0.078	0.036
Click	**0.022**	0.007	0.010	0.013	**0.095**	0.013	0.032	0.019
Covertype	0.148	**0.162**	0.059	0.073	**0.233**	0.180	0.065	0.136
CPU	**0.149**	0.001	0.004	0.011	**0.038**	0.009	0.009	0.002
Diabetes	**0.109**	0.090	0.073	0.061	**0.210**	0.102	0.064	0.008
FMNIST	0.049	**0.056**	0.040	0.035	**0.083**	0.072	0.021	0.017
MNIST-2	0.005	0.004	0.013	**0.016**	0.004	0.002	0.005	**0.010**
MNIST-10	0.201	**0.211**	0.155	0.053	**0.191**	0.145	0.105	0.056
Phoneme	**0.135**	0.014	0.026	0.046	0.042	0.047	0.048	**0.090**

Table 10: Weighted accuracy drop for transferring SVM-RBF values to logistic regression and KNN using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Logistic Regression				KNN			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.084**	0.078	0.053	0.071	0.101	**0.104**	0.083	0.056
Click	0.006	0.008	**0.010**	0.007	**0.005**	0.004	0.004	0.002
Covertype	**0.228**	0.199	0.084	0.174	**0.245**	0.171	0.111	0.106
CPU	0.017	0.017	**0.038**	0.007	0.045	**0.047**	0.041	-0.011
Diabetes	0.067	0.079	**0.096**	0.069	**0.166**	0.073	0.053	0.019
FMNIST	**0.060**	0.035	0.035	0.054	**0.050**	0.032	0.021	0.004
MNIST-2	0.007	0.006	0.007	**0.010**	0.004	**0.007**	0.005	0.005
MNIST-10	0.110	**0.131**	0.061	0.065	0.166	**0.232**	0.089	0.066
Phoneme	**0.060**	0.004	0.011	-0.032	**0.044**	0.029	0.035	0.009

Table 11: Weighted accuracy drop for transferring SVM-RBF values to gradient boosting classifier and multi-layer perceptron (MLP) using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Gradient Boosting				MLP			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	0.101	**0.126**	0.050	0.094	**0.130**	0.093	0.079	0.071
Click	0.007	0.006	**0.009**	0.007	**0.030**	0.013	0.021	-0.002
Covertype	0.138	**0.139**	0.056	0.096	**0.210**	0.142	0.083	0.119
CPU	0.011	0.008	**0.021**	0.007	0.008	**0.018**	0.014	-0.001
Diabetes	0.062	**0.086**	0.051	0.036	**0.180**	0.104	0.047	0.109
FMNIST	**0.075**	0.036	0.047	0.025	**0.098**	0.027	0.031	0.062
MNIST-2	0.008	0.011	0.012	**0.018**	**0.012**	0.008	0.002	0.005
MNIST-10	0.226	**0.263**	0.125	0.104	**0.165**	0.159	0.081	0.091
Phoneme	**0.045**	0.041	0.013	0.016	**0.055**	0.044	0.046	0.017

Table 12: Weighted accuracy drop for transferring KNN values to logistic regression and SVM-RBF using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Logistic Regression				SVM-RBF			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	0.014	0.033	**0.035**	0.014	0.040	0.051	**0.057**	0.018
Click	0.002	0.006	**0.132**	0.007	0.004	0.004	0.004	0.004
Covertype	**0.173**	0.163	0.100	0.091	0.160	**0.174**	0.067	0.109
CPU	0.006	0.011	**0.017**	0.006	**0.044**	0.007	0.007	0.007
Diabetes	0.070	0.080	**0.107**	0.089	**0.118**	0.051	0.009	0.036
FMNIST	0.021	0.026	0.027	**0.046**	0.024	0.027	0.029	**0.045**
MNIST-2	0.010	0.010	0.009	**0.013**	0.011	**0.013**	0.010	0.010
MNIST-10	0.023	**0.060**	0.052	0.059	0.087	**0.149**	0.077	0.109
Phoneme	**0.027**	0.005	0.003	0.017	**0.049**	0.040	0.029	0.010

Table 13: Weighted accuracy drop for transferring KNN values to gradient boosting classifier and multi-layer perceptron (MLP) using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

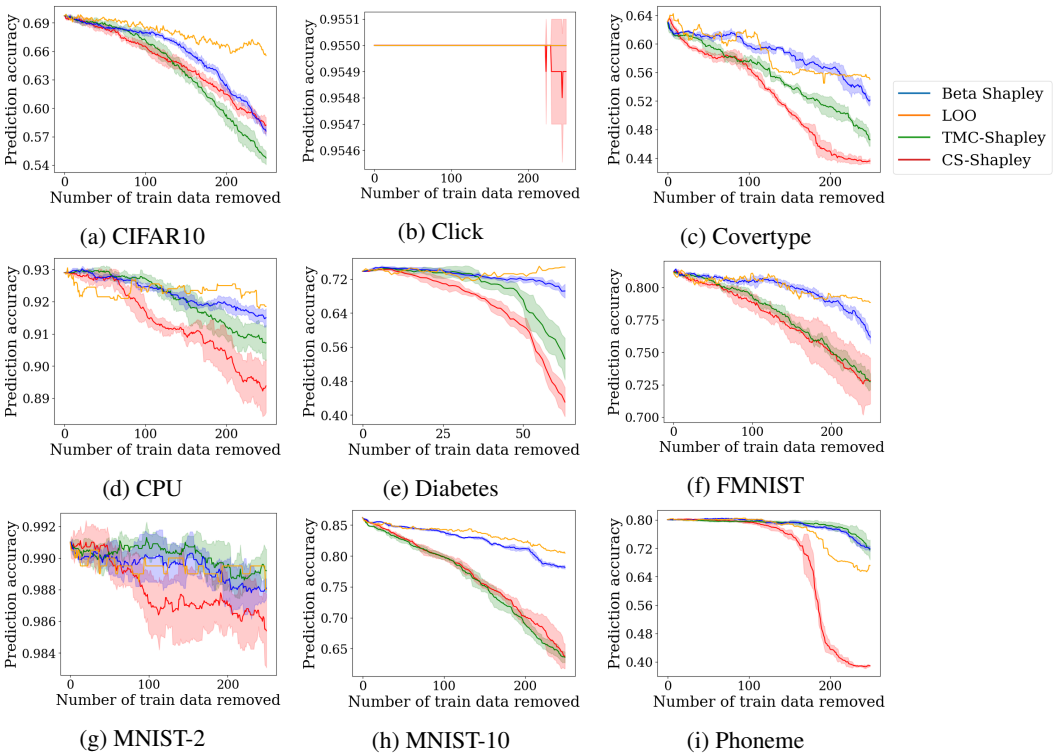
Dataset	Gradient Boosting				MLP			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	0.044	0.039	**0.062**	0.032	**0.040**	0.038	0.016	0.002
Click	0.005	0.006	0.009	0.009	0.025	0.012	**0.043**	-0.008
Covertype	**0.124**	0.109	0.046	0.032	0.179	**0.191**	0.124	0.080
CPU	**0.015**	0.004	0.006	0.001	**0.027**	0.0001	0.012	-0.0001
Diabetes	**0.084**	0.061	0.065	0.069	**0.188**	0.093	0.077	0.134
FMNIST	0.042	0.013	**0.043**	0.025	0.032	0.044	0.025	**0.056**
MNIST-2	0.009	0.003	0.010	**0.015**	0.008	0.006	0.008	0.008
MNIST-10	0.131	**0.164**	0.096	0.147	0.077	0.080	0.098	**0.105**
Phoneme	0.033	**0.036**	0.021	0.009	0.053	0.038	**0.064**	0.055

Table 14: Weighted accuracy drop for transferring gradient boosting classifier values to logistic regression and SVM-RBF using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	Logistic Regression				SVM-RBF			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.091**	0.040	0.062	0.020	**0.073**	0.048	0.070	0.028
Click	0.011	0.006	**0.014**	0.003	0.004	0.004	0.004	0.004
Covertype	**0.143**	0.141	0.070	0.026	0.095	**0.103**	0.084	0.047
CPU	0.006	**0.012**	0.004	0.007	**0.021**	0.013	0.010	0.005
Diabetes	**0.097**	0.070	0.089	0.009	**0.084**	0.051	0.009	-0.044
FMNIST	**0.060**	0.033	0.038	0.021	**0.050**	0.033	0.023	0.021
MNIST-2	0.008	0.006	0.009	0.009	0.004	0.004	0.003	**0.006**
MNIST-10	–	–	–	–	–	–	–	–
Phoneme	**0.013**	-0.002	0.002	0.001	**0.042**	0.029	0.033	0.020

Table 15: Weighted accuracy drop for transferring gradient boosting classifier values to KNN and multi-layer perceptron (MLP) using CS-SHAPLEY (CS), TMC-Shapley (TMC), Beta Shapley (Beta), and Leave-One-Out (LOO).

Dataset	KNN				MLP			
	CS	TMC	Beta	LOO	CS	TMC	Beta	LOO
CIFAR10	**0.067**	0.056	0.061	0.019	0.033	**0.114**	0.019	0.060
Click	**0.009**	0.005	0.004	0.003	**0.035**	0.021	0.022	-0.009
Covertype	0.089	**0.114**	0.071	0.053	0.138	**0.222**	0.023	-0.008
CPU	**0.019**	0.005	0.009	0.017	**0.017**	0.008	0.008	0.004
Diabetes	**0.117**	0.087	0.029	0.022	**0.178**	0.110	0.063	0.071
FMNIST	**0.037**	0.023	0.017	0.010	**0.075**	0.028	0.047	0.021
MNIST-2	**0.007**	0.005	0.005	0.006	**0.007**	0.006	0.002	0.006
MNIST-10	–	–	–	–	–	–	–	–
Phoneme	**0.044**	0.028	0.034	0.022	**0.080**	0.058	0.068	0.025

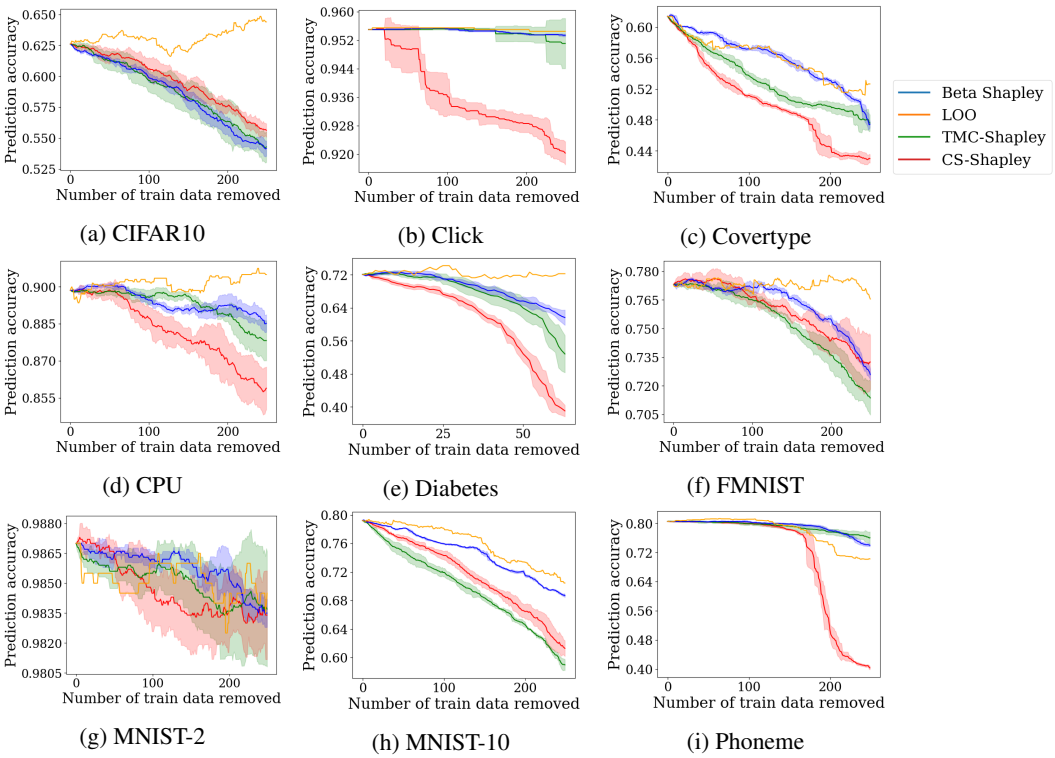


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 10: Performance across datasets when transferring values from logistic regression to SVM-RBF for the high-value removal task.

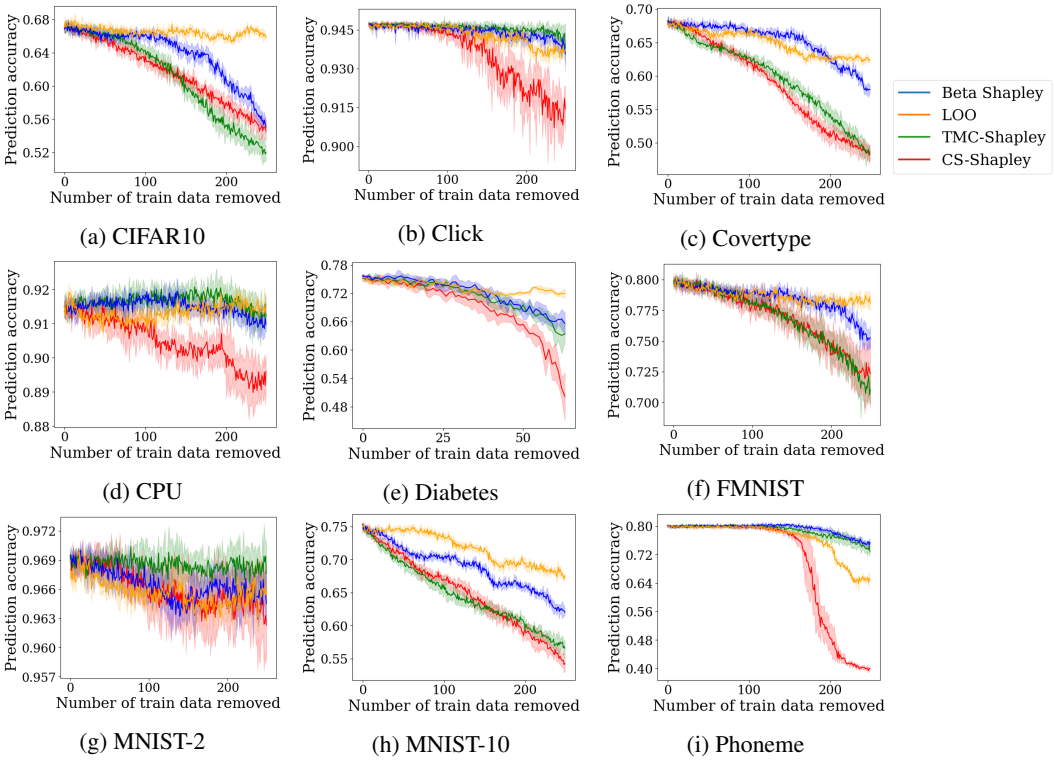


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 11: Performance across datasets when transferring values from logistic regression to KNN for the high-value removal task.

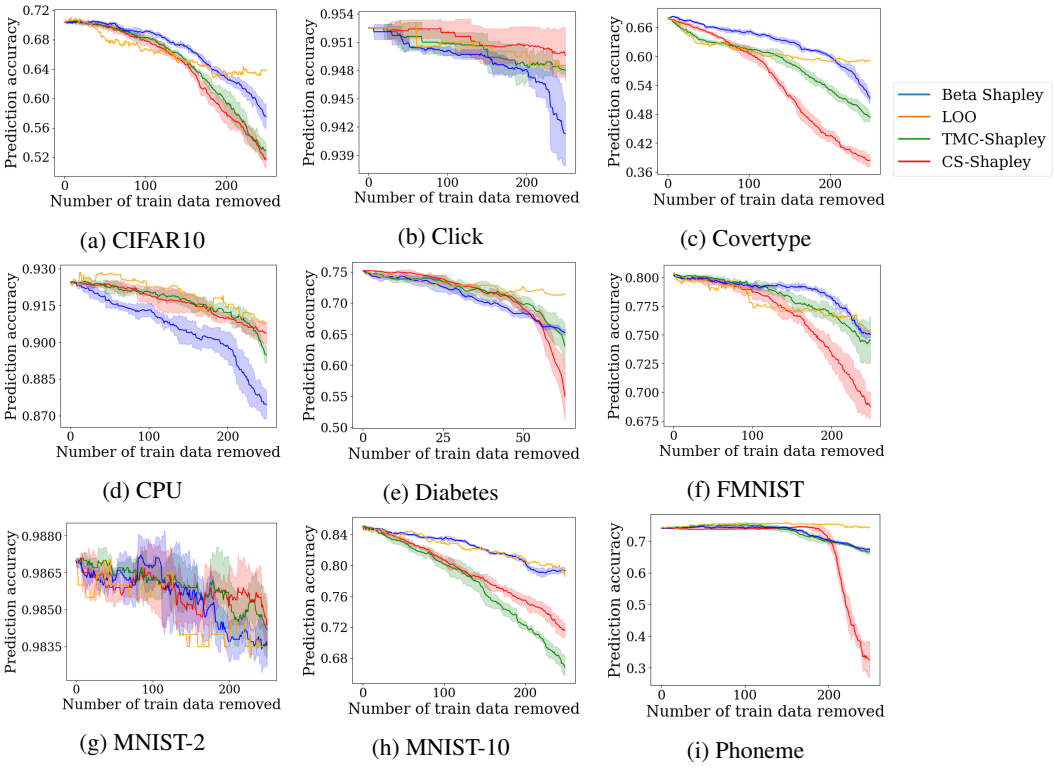


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 12: Performance across datasets when transferring values from logistic regression to gradient boosting classifier for the high-value removal task.

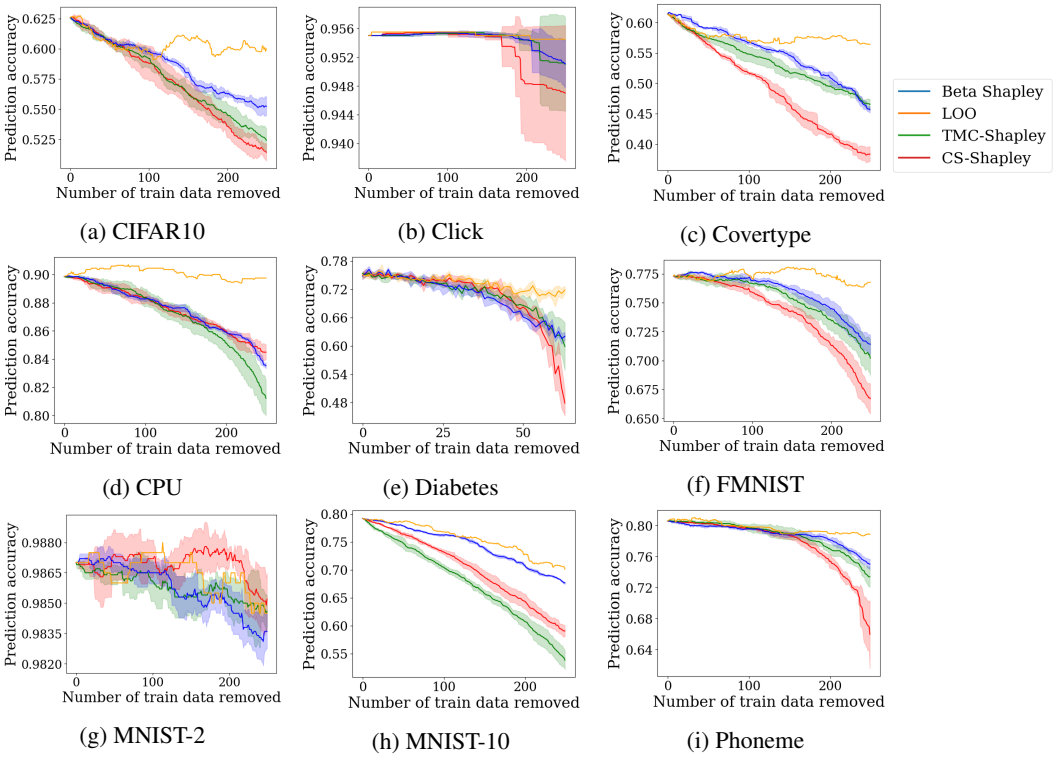


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 13: Performance across datasets when transferring values from SVM-RBF to logistic regression for the high-value removal task.



(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 14: Performance across datasets when transferring values from SVM-RBF to KNN for the high-value removal task.

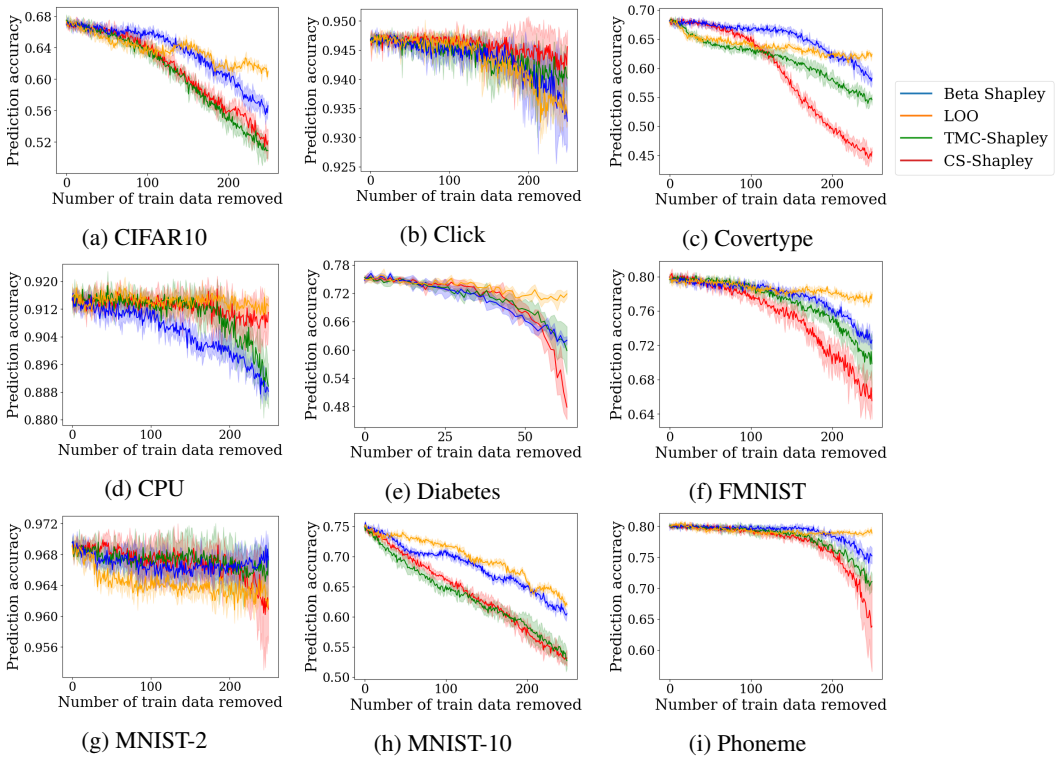


Figure 15: Performance across datasets when transferring values from SVM-RBF to gradient boosting classifier for the high-value removal task.

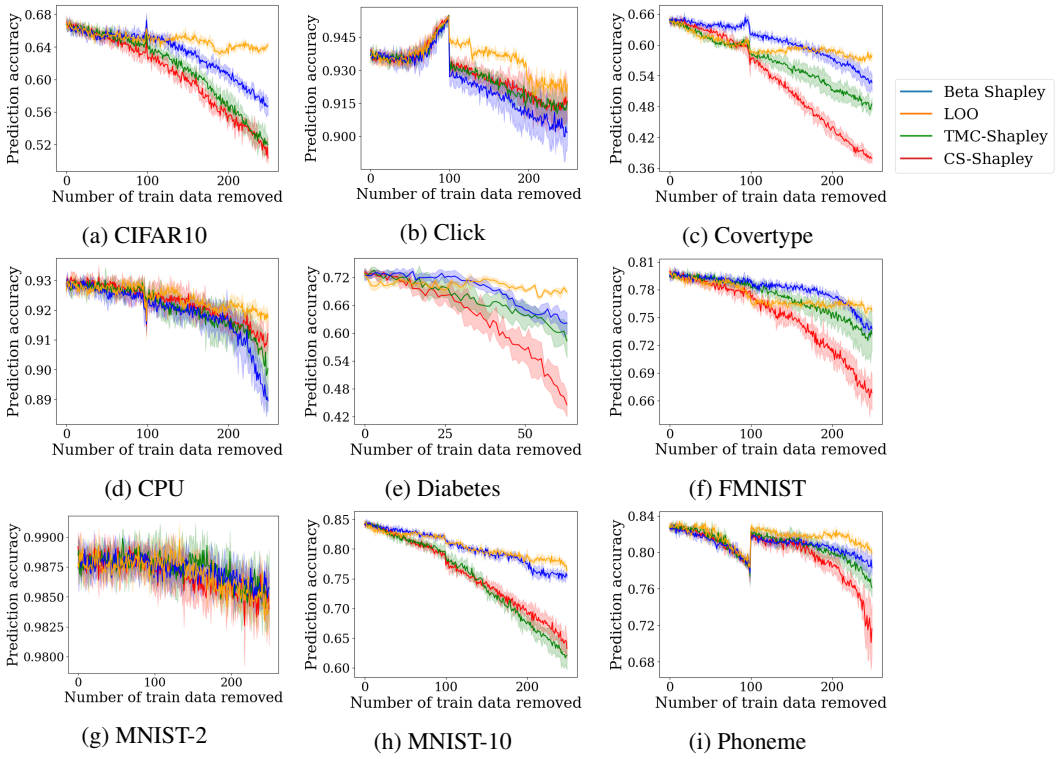


Figure 16: Performance across datasets when transferring values from SVM-RBF to MLP for the high-value removal task.

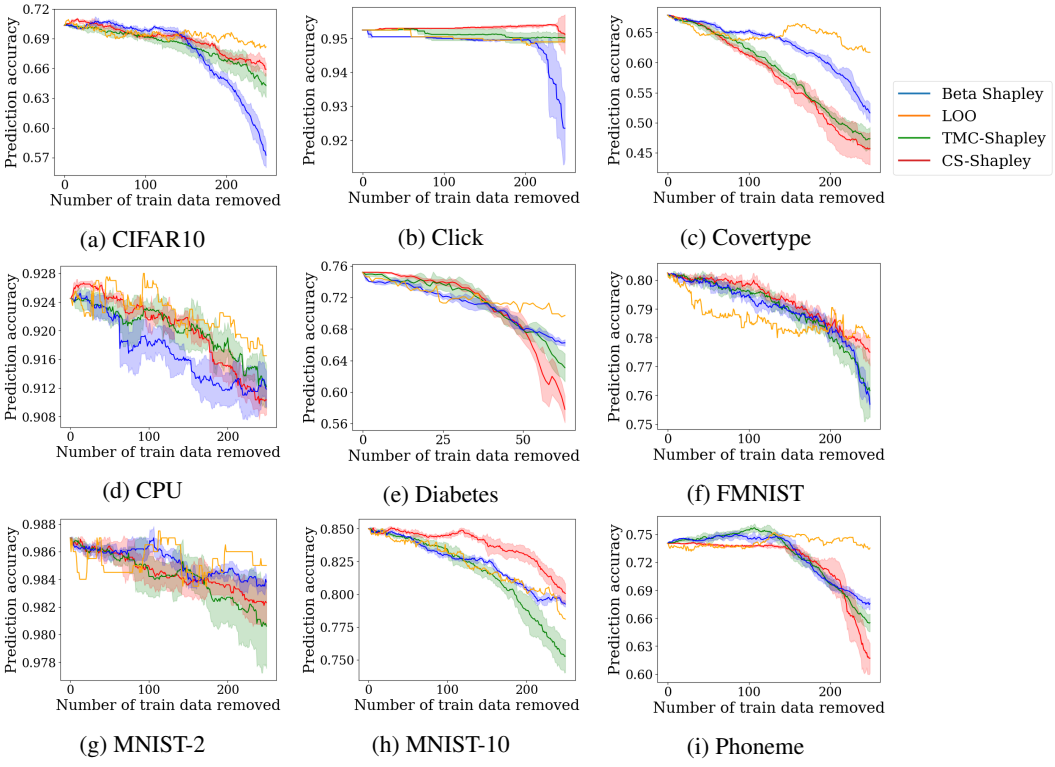


Figure 17: Performance across datasets when transferring values from KNN to logistic regression for the high-value removal task.

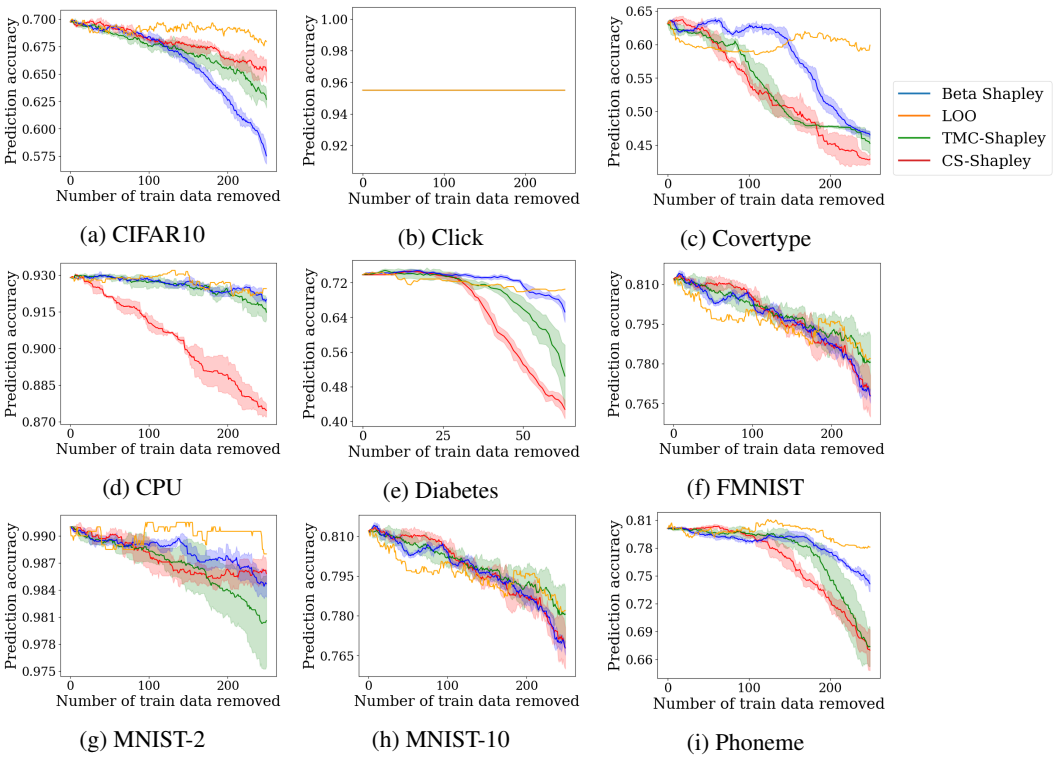
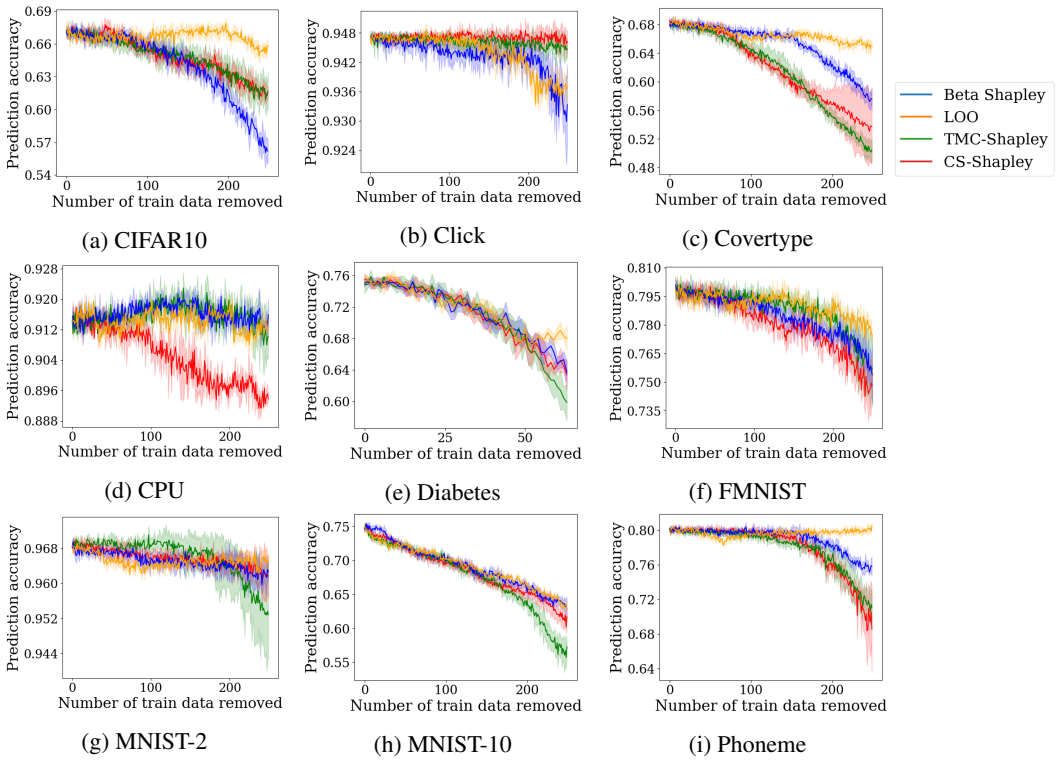


Figure 18: Performance across datasets when transferring values from KNN to SVM-RBF for the high-value removal task.

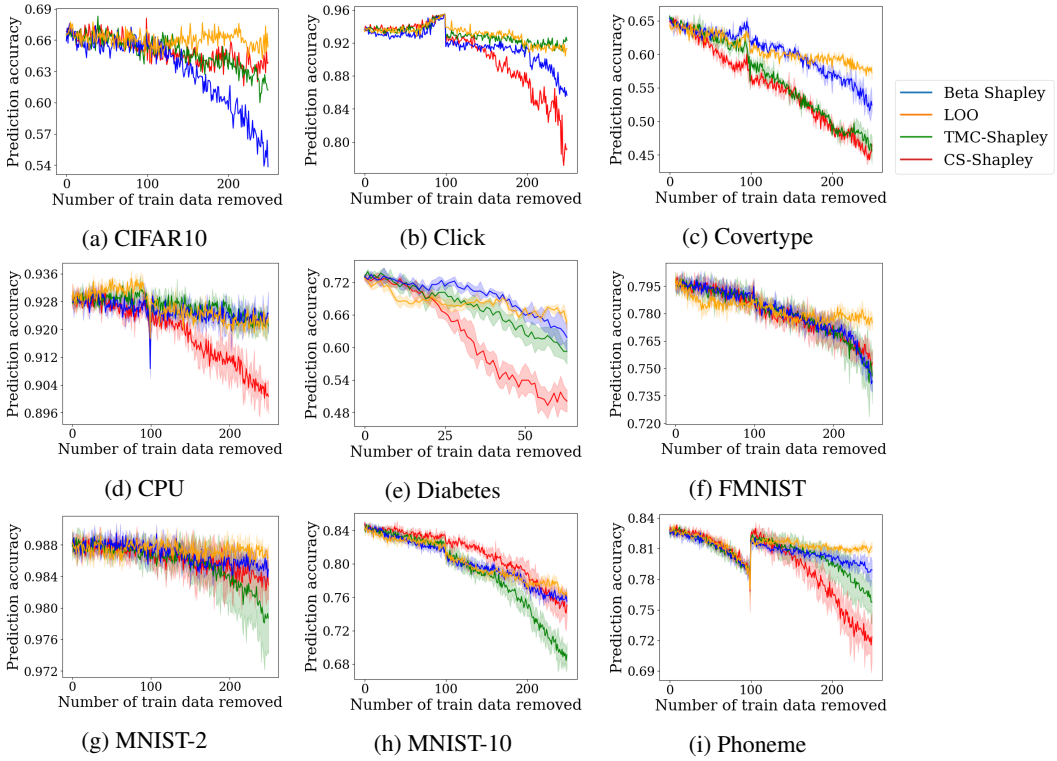


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 19: Performance across datasets when transferring values from KNN to gradient boosting classifier for the high-value removal task.

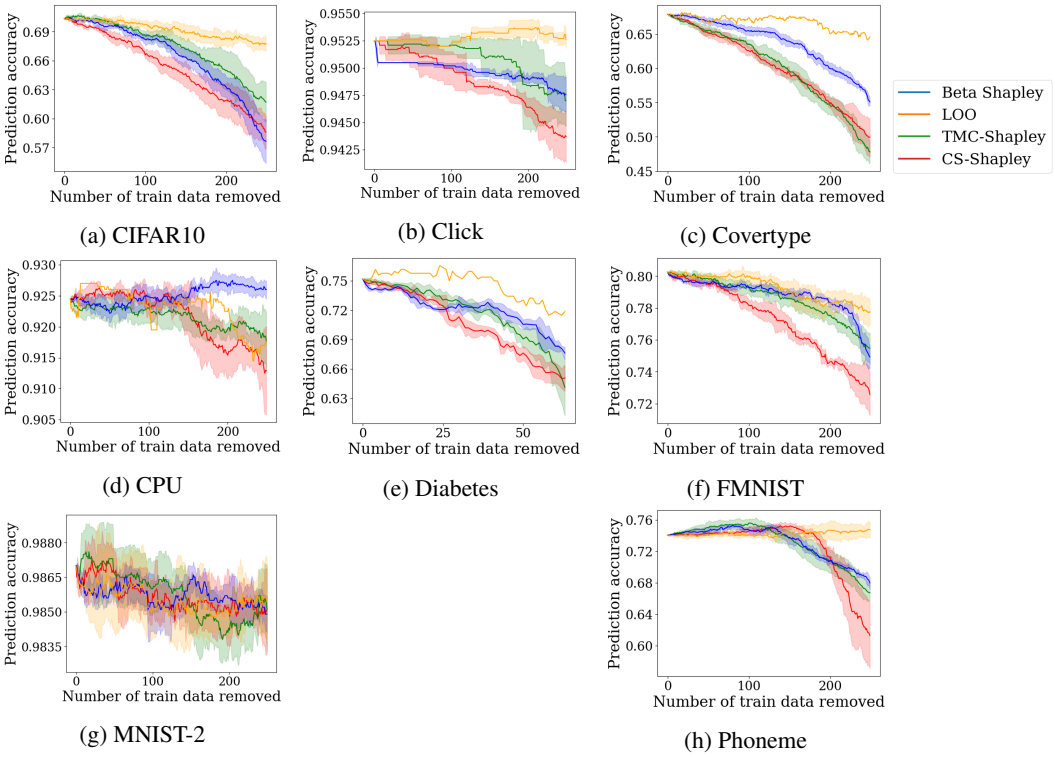


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) MNIST-10 (i) Phoneme

Figure 20: Performance across datasets when transferring values from KNN to MLP for the high-value removal task.

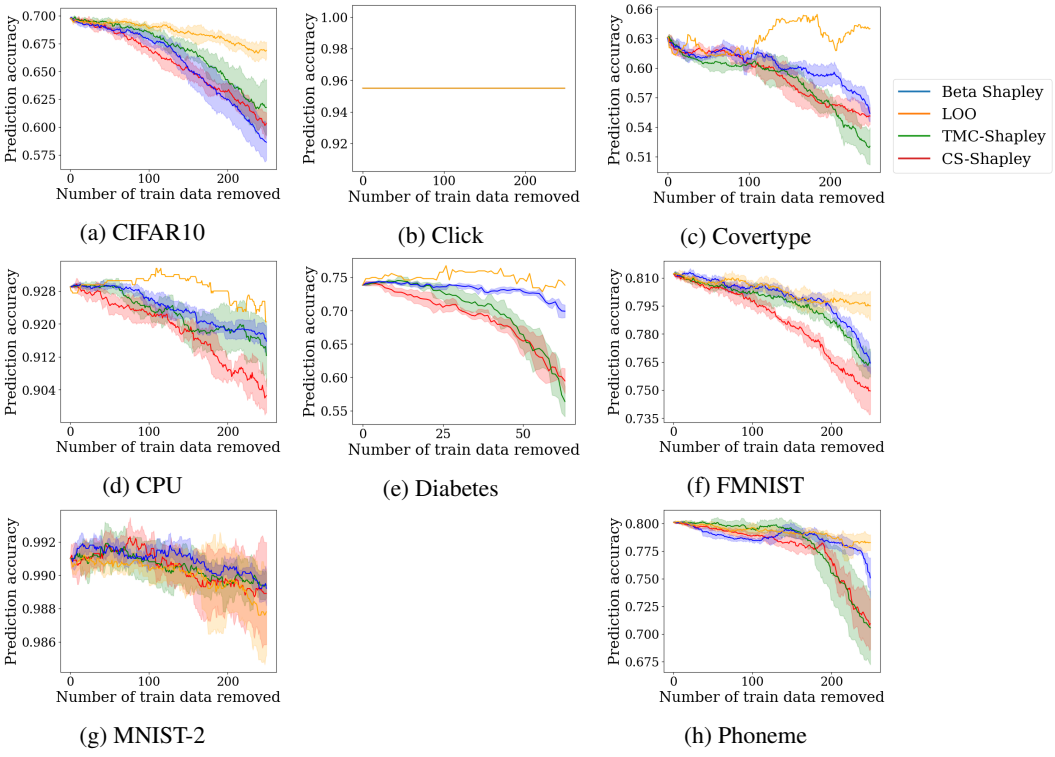


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) Phoneme

Figure 21: Performance across datasets when transferring values from gradient boosting classifier to logistic regression for the high-value removal task.

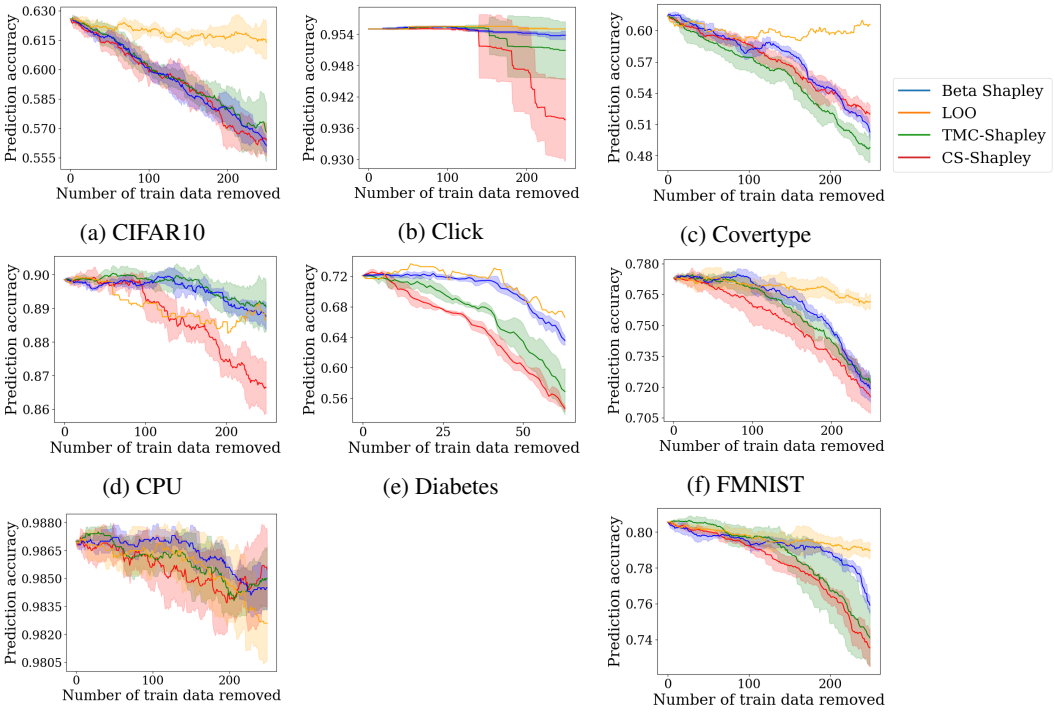


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) Phoneme

Figure 22: Performance across datasets when transferring values from gradient boosting classifier to SVM-RBF for the high-value removal task.

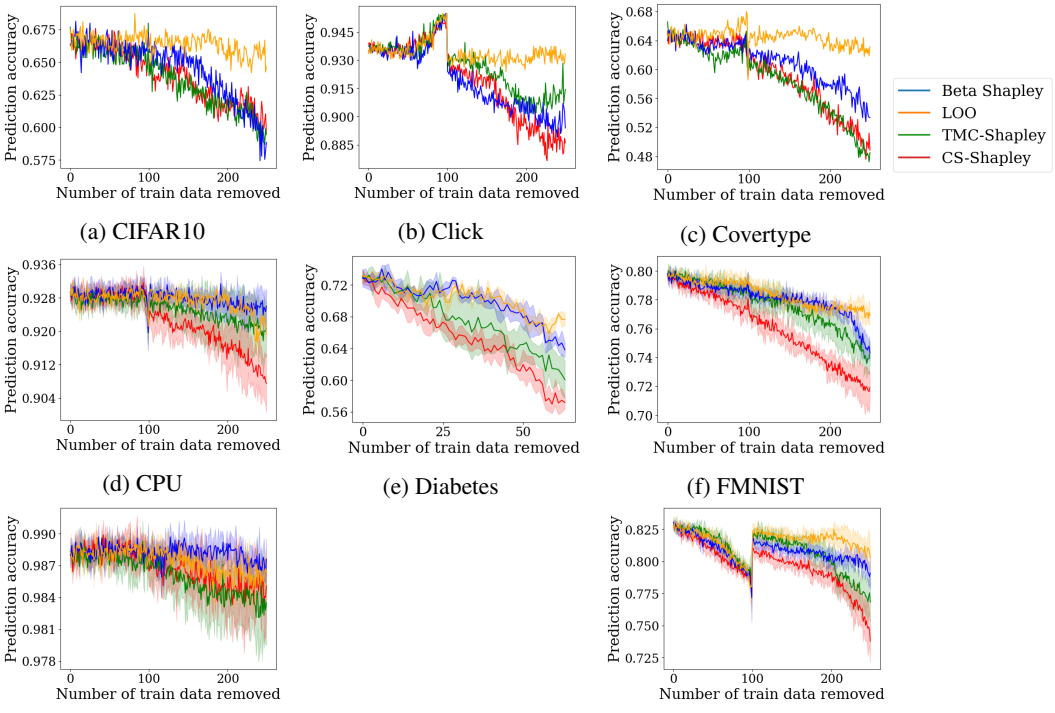


(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) Phoneme

Figure 23: Performance across datasets when transferring values from gradient boosting classifier to KNN for the high-value removal task.



(a) CIFAR10 (b) Click (c) Covertype

(d) CPU (e) Diabetes (f) FMNIST

(g) MNIST-2 (h) Phoneme

Figure 24: Performance across datasets when transferring values from gradient boosting classifier to MLP for the high-value removal task.