APPENDIX A

Due to lack of space, we provide additional results in this Appendix, which is provided as supplemental material to the revisions.

A.1 Scenario 6: Interpreting SVM vs. DT

Setup Scenario 6. We use the following setup from Table 3: *Dataset:* NoMoAds. *Users:* None. *Classifier Granularity:* General. *Models:* Centralized SVM vs. DT. *Tasks:* PII exposure. Prior work chose DT over other models partially because of their interpretability. In our context, these models learn similar separation of our datasets, which we demonstrate by (1) observing the most important coefficients in SVM, (2) by knowledge transfer from SVM to DT. The goal here is to compare SVM to DT in terms of their interpretability.

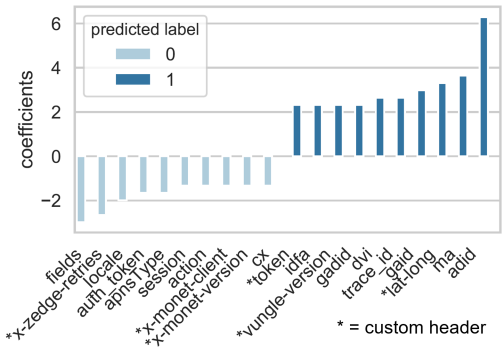
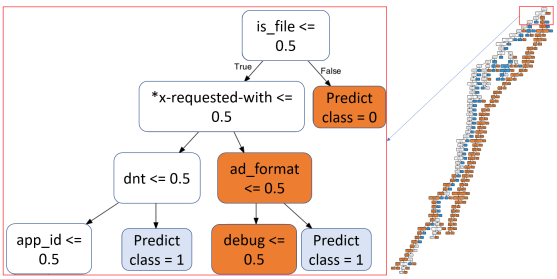
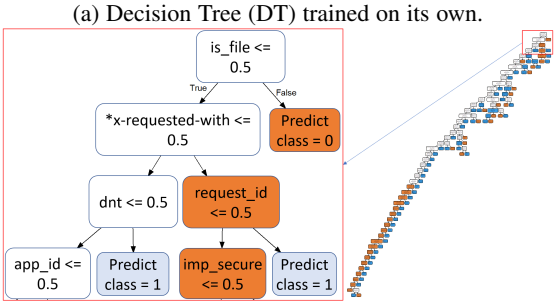


Fig. 20: Top 10 negative and positive coefficients and the corresponding features obtained from Centralized SVM.



(a) Decision Tree (DT) trained on its own.



(b) First train SVM, then transfer knowledge to DT.

Fig. 21: Interpretability of DT vs. SVM in Setup 5.

Results Scenario 6. Fig. 20 shows the ten most important negative and positive coefficients and their corresponding features for our Centralized SVM. In order to distinguish important features, we use the model's coefficients, where the positive ones correspond to the features whose presence leads to positive labels and the negative coefficients correspond to features responsible for predicting label 0 (*e.g.,* No PII detected). This is not a one-to-one mapping of important features between SVM and DT due to their internal representation of features. However, we observe certain keys that are responsible for PII exposures such as "gaid", that also appear in the corresponding DT.

Fig. 21(b) shows the DT after knowledge transfer from SVM. To perform knowledge transfer from SVM to DT, we first split the data into 40% for training SVM, another 40% for training a DT, which is labeled with predictions from the aforementioned SVM. The remaining 20% of the data is used for testing. This is one way to leverage the interpretability of DTs via knowledge transfer from SVM. In Fig. 21(a), we show a DT which was trained with NoMoAds for PII prediction, while in Fig. 21(b), we show the DT after knowledge transfer from SVM. We observe that both DTs, at least at the top levels, have similar important features and thus, capture similar patterns. The original DT and SVM reached F1 score = 0.95 and the after knowledge transfer DT reached F1 score = 0.94 on the same test data. This is only a minor F1 score loss during knowledge transfer.

The most notable difference between the trees in Fig. 21 is the lack of a large branch that only predicts label 0, which is the result of how the original tree unsuccessfully attempts to separate data. However, the DT after the knowledge transfer is oblivious to this error, since the SVM most likely suffers from the same issue as the original DT. Such errors propagating from the SVM make the DT after the knowledge transfer smaller (269 vs. 141 nodes) than the original DT. Please refer to the appendix of the extended version of this work [40] for full pictures of the above DTs.

A.2 Additional Results for Scenario 2: Comparison of Training Time.

This section of the Appendix provides additional results extending Sec. 5.2.

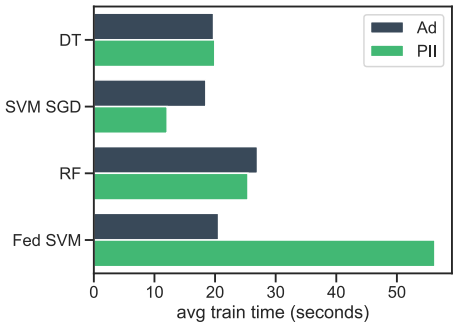


Fig. 22: **Results 2c.** Comparison of average (from 5 runs) training time for Centralized models with NoMoAds data using HTTP Keys and Federated SVM with 20 synthetic users for both prediction tasks.

Results 2c: Training Time. Fig. 22 depicts the average training time (from 5 runs) of the baseline centralized models with NoMoAds data for both prediction tasks in comparison to Federated SVM with 20 even synthetic users.[7] To measure the time for the Federated models, we set the same target F1 score as before (0.95 for PII and 0.85 for Ads) and we report the total

7. Time was measured on a machine with Intel(R) Xeon(R) CPU E5-2623 v3 @ 3.00GHz and 62GB RAM. The reported train times are on models with default parameters as selected in scikit-learn [68], except for Random Forest (RF) which we limit to 25 estimators instead of the default 100 estimators.

training time from all rounds required until the convergence with E=1, B=10, C=1.0. We define the Federated training time as: $t_{fed} = max(t_{local}) + t_{aggr}$ and thus, it depends on the worst case local train time. The aggregation time on the server side, t_{aggr}, is negligible as it takes only 40 microseconds. Overall, the SVM with SGD is comparable to DT and RF and the Federated SVM is comparable to the centralized models; PII task requires 3 rounds (each round taking approximately 145 seconds), while for the Ads task one round was sufficient resulting in faster training.

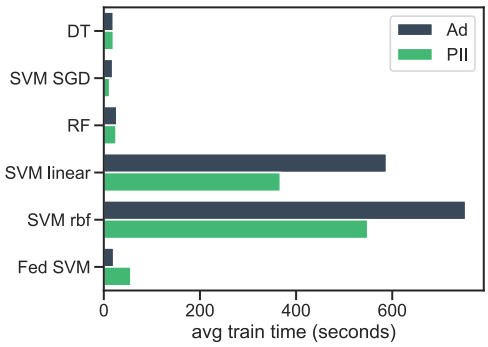


Fig. 23: Comparison of average training time for all Centralized models from Table 4 with NoMoAds data using HTTP Keys and Federated SVM with 20 synthetic users for both prediction tasks.

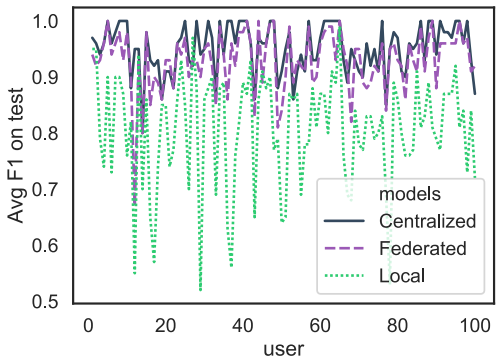
A.3 Additional Results on the Benefit of Federation

This section of the Appendix extends Sec. 5.3 using Setup 3b with 100 synthetic users and uneven IID split of data. It continues the "Results 4e" from Section 5.4, "Benefit of Crowdsourcing", but considers 100 synthetic users instead of 10 real non-IID users and it compares the performance of Federated to Centralized and Local models. The goal is to further demonstrate the benefit of moving away from locally trained models, to crowdsourced models, especially Federated (which essentially performs as well as a Centralized model).

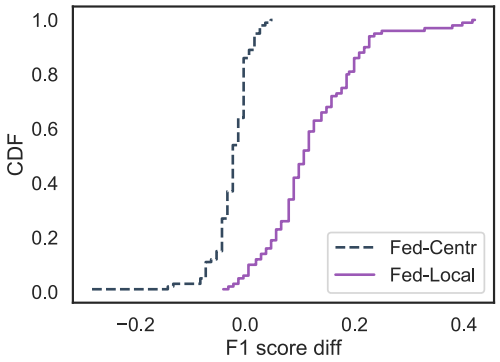
Fig. 24 shows the results for this scenario. We compare the F1 score on each user's test data for crowdsourced models: Centralized (their training data was shared with a server) and Federated (only model parameters were shared), and their corresponding Local models (they shared nothing). Fig. 24(a) shows the F1 score achieved from Federated, Centralized and locally trained models for each of the 100 users when tested on each user's test data. Fig. 24(b) shows the empirical CDF of the difference between the F1-score of Federated and Centralized models and between the Federated and Local models. We make two observations by looking at Fig. 24(b). First, the Federated achieves similar F1 score to the Centralized model for 90% of the users, except for a few users, that Centralized performs slightly better. Second, the Federated model performs better than the corresponding Local models: for 80% of users the Federated F1 score reaches an increase up to 0.2, compared to the Local model, and for some users it is almost 0.4, which is significant. In summary, all users benefit from the use of crowdsourcing, *i.e.,* there are positive differences in F1 score for (Federated-Local), but at different degrees.

A.4 Additional Results on Threat Setup 6b.

This section of the Appendix provides additional results extending Section 6.1 on Threat Setup 6b. Fig. 25 shows the CDFs of per-domain prediction with HTTP Keys and Recon Words. Fig. 26 shows the per-domain F1 score separating the ATS from non-ATS domains and comparing the performance with HTTP Keys and Recon Words for all 105 domains. Fig. 27 shows the results for a zoomed-in version of the top 30 domains (sorted alphabetically).



(a) Raw values of F1 score of Federated, Centralized and Local models when tested on each user's test data.



(b) Empirical cumulative distribution function (CDF) of the differences of F1 scores for (Federated − Local) vs. (Federated − Centralized).

Fig. 24: Comparison of Local vs. Federated vs. Centralized models when tested on each of the 100 uneven synthetic users with AntShield data. Federated F1 score is comparable to Centralized and both perform better (positive difference in F1) than the corresponding Local models. All users benefit from the crowdsourced models due to IID nature of the data, but at a different degree: the increase in F1 can be up to 0.4, with 80% of the users up to 0.2.

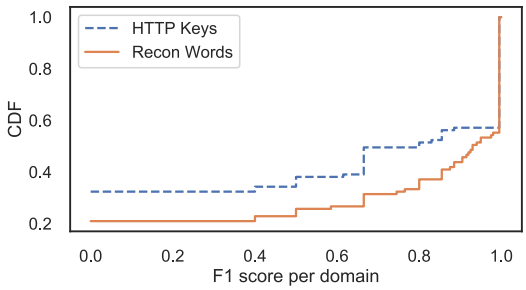
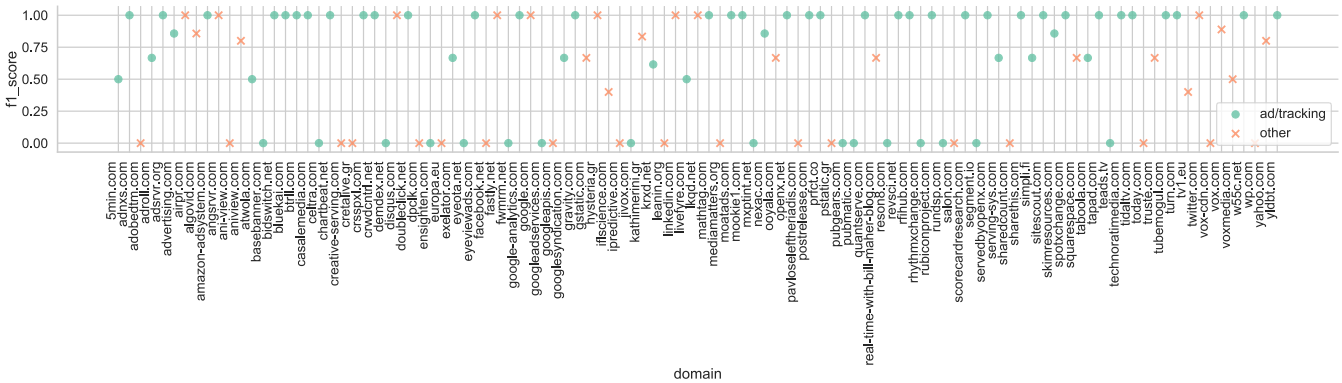
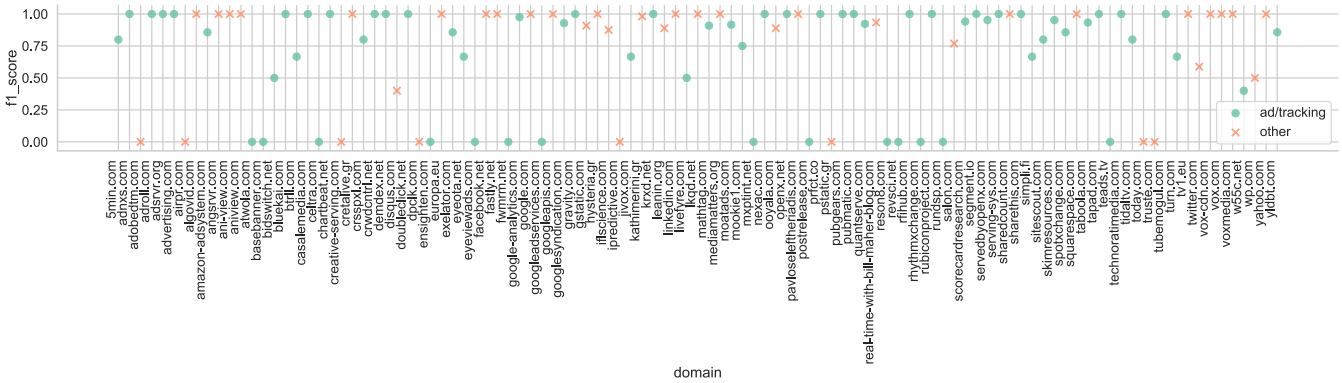


Fig. 25: Empirical cumulative distribution function (CDF) for per-domain F1 score with HTTP Keys compared to Recon Words.

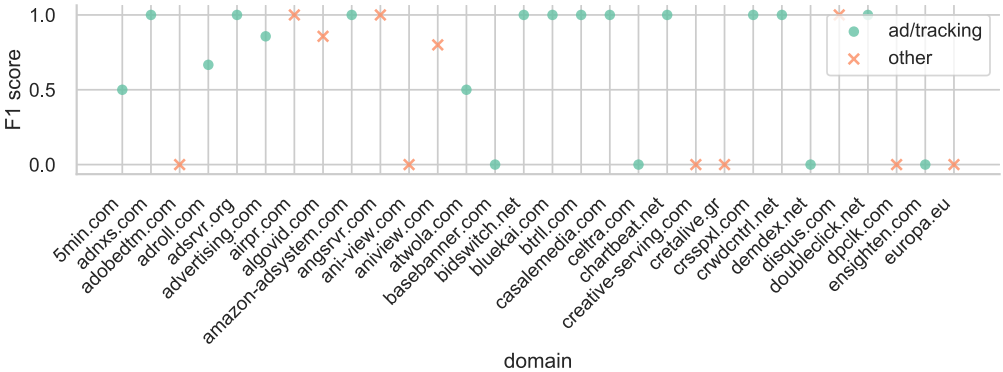


(a) Per-domain F1 score with HTTP Keys

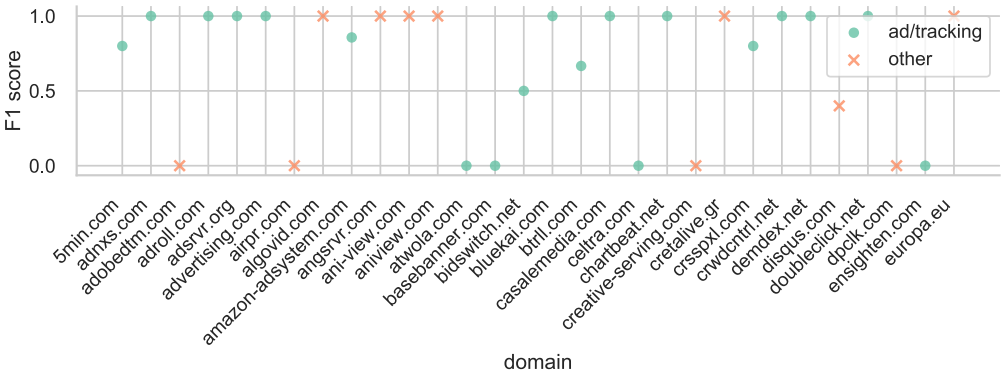


(b) Per-domain F1 score with Recon Words

Fig. 26: Comparison of per-domain prediction with SVM and two feature spaces: Recon Words and HTTP Keys for all 105 domains. "Advertising and Tracking" (ATS domains), marked with "o", are usually contacted by third party libraries used by mobile apps, and are thus less sensitive. "Other" (non-ATS) domains, marked with "x", reflect the domains the user actually intended to visit and are more sensitive.



(a) Per-domain F1 score with HTTP Keys



(b) Per-domain F1 score with Recon Words

Fig. 27: Zoomed-in version of per-domain prediction with SVM and two feature spaces: Recon Words and HTTP Keys.

Algorithm 2: Attack with Secure Aggregation in place

Given a target user t, K total number of clients, k number of clients who participate in a round, n feature size, encoded_value_certain value that represents a feature is present with certainty:
Initialize feature_array[K][n]=-1
$S_{feat} \leftarrow$ (set of features seen so far)

Phase 1:
$S_{tuples} \leftarrow$ (set of all user k-tuples with t being the first user in each tuple)
for *each k-tuple* $p \in S_{tuples}$ **do**
  $S_{cur} \leftarrow$(union of features of users in p)
  if S_{feat} *is empty* **then**
    for *each* u *in* p **do**
      for *each feature* $f \in S_{cur}$ **do**
        feature_array[u][f] +=1
  else
    $S_{new} \leftarrow Scur - S_{feat}$
    for *each feature* $f \in S_{new}$ **do**
      feature_array[p[0]][f] = 0
      if *len(p)==2* **then**
        feature_array[p[1]][f] = encoded_value_certain
      else
        for *each* u *in p[1:]* **do**
          feature_array[u][f] +=1
    $S_{common} \leftarrow Scur \cap S_{feat}$
    for *each feature* $f \in S_{common}$ **do**
      for *each* u *in* p **do**
        feature_array[u][f] +=1
    $S_{prev} \leftarrow S_{feat} - Scur$
    for *each feature* $f \in S_{prev}$ **do**
      for *each* u *in* p **do**
        feature_array[u][f] = 0
      for *each* u *in all_users* **do**
        if *feature_array[u][f]* ≥ 1 *&&* *feature_array[u][f]!=encoded_value_certain* **then**
          feature_array[u][f] +=1
  S_{feat}.update(S_{cur})

Phase 2:
$S_{feat2} \leftarrow$ (set of features seen in Phase 2)
for u *in set(K-t)* **do**
  $S_{tuples2} \leftarrow$ (set of user k-tuples (without the target t) with u being the first user in each tuple)
  for *each k-tuple* $p \in S_{tuples2}$ **do**
    repeat phase 1 with u as target
    S_{feat2}.update(S_{cur})

After Phase 2:
for f *in* S_{feat} - S_{feat2} **do**
  feature_array[t][f] = encoded_value_certain
for f *in* S_{feat2} - S_{feat} **do**
  feature_array[t][f] = 0
for u *in range(K)* **do**
  for f *in feature_array[u]* **do**
    if *feature_array[u][f] != 0* *feature_array[u][f] != encoded_value_certain* **then**
      feature_array[u][f] = feature_array[u][f]/rounds_user_u_seen

A.5 Additional Evaluation of Mitigation Approach

A5.1 This section of the Appendix provides additional results extending Section 6.2 on Mitigation. Algorithm 2 shows the details of the algorithm used in Sec. 6.2. Fig. 28 shows the accuracy of the privacy attack with Secure Aggregation on. Moreover, we show more statistics about the participating users from in-house Facebook dataset regarding their feature (HTTP Keys) size (Fig. 29) and their pairwise similarity based on common features (HTTP Keys) (Fig. 30) .

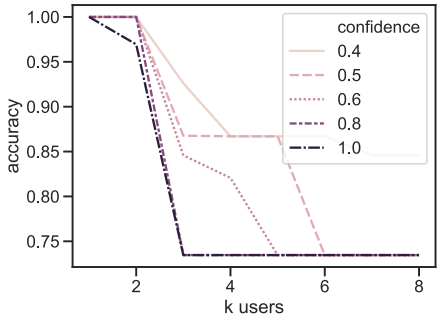


Fig. 28: Evaluating Attack Algorithm 6c, with secure aggregation on. We report the accuracy ($\frac{TP+TN}{T+P}$), for varying k (participating users in a round) and confidence threshold.

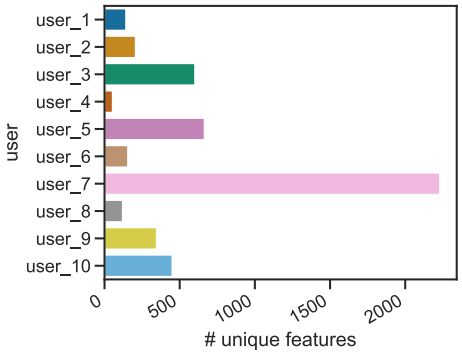


Fig. 29: Per user unique HTTP Keys features for in-house Facebook dataset.

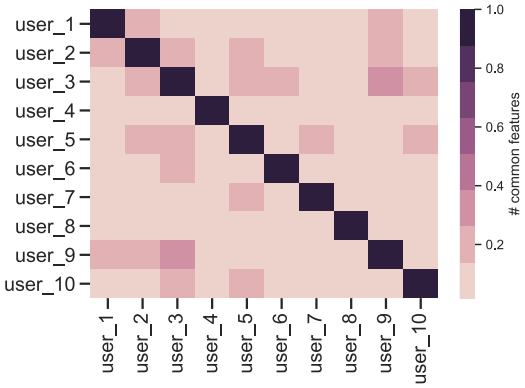
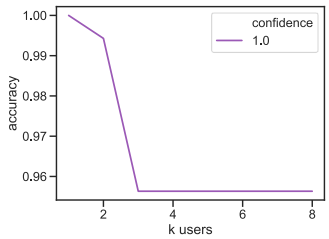
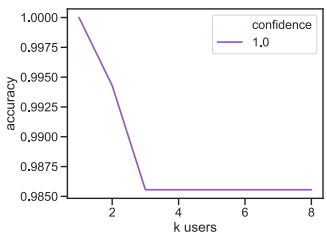


Fig. 30: Pairwise user similarity based on Jaccard similarity of common features for in-house Facebook dataset.

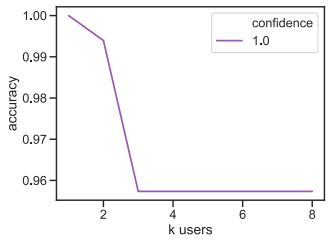
A5.2: Targeting other users. This section of the Appendix provides additional results extending Section 6.2 on Mitigation when other users are targeted instead of User 7.



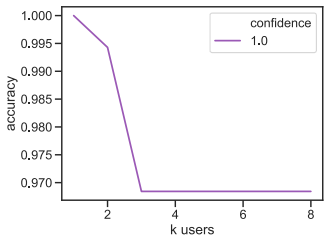
(a) Target is User 1.



(b) Target is User 4.

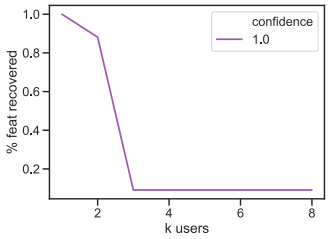


(c) Target is User 6.

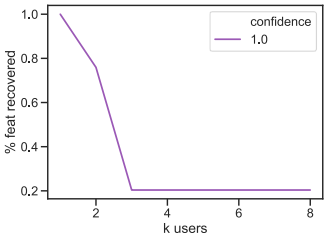


(d) Target is User 8.

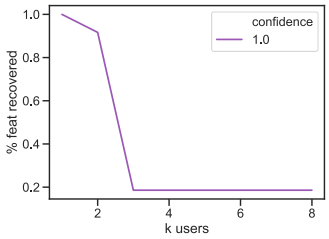
Fig. 31: Accuracy of recovered features for different target users from Facebook dataset with maximum confidence. The fewer the unique features a user has (user 4 has the fewest), the better the worst case accuracy is for features recovered with maximum confidence.



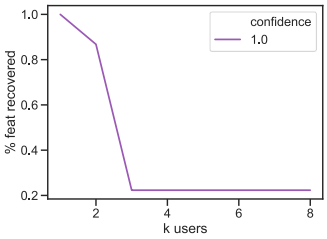
(a) Target is User 1.



(b) Target is User 4.

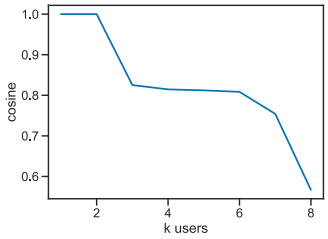


(c) Target is User 6.

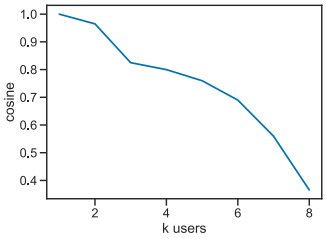


(d) Target is User 8.

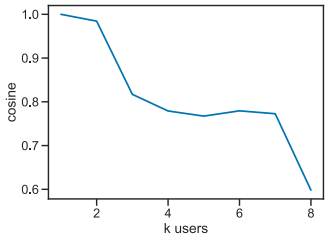
Fig. 32: Percent of recovered features for different target users from Facebook dataset with maximum confidence.



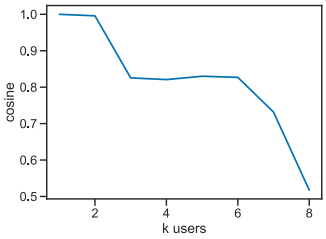
(a) Target is User 1.



(b) Target is User 4.



(c) Target is User 6.



(d) Target is User 8.

Fig. 33: Cosine similarity of true and recovered features for different target users from Facebook dataset. The larger the k the worse the cosine similarity is due to decreasing confidence levels which increases the distance between true and recovered features.