Black Box Attacks on Machine Learning Model and Strategies to Mitigate them

Shreeraj Shah
sbs577

Prachi Gupta
pg1647

https://github.com/pg1647/intromlproject

Abstract:

Machine learning(ML) models today are vulnerable to several types of attacks. In this work, we will study a category of attack known as membership inference attack and show how ML models are susceptible to leaking secure information under such attacks. Given a data record and a black box access to a ML model, we present a framework to deduce whether the data record was part of the model's training dataset or not. We achieve this objective by creating an attack ML model which learns to differentiate the target model's predictions on its training data from target model's predictions on data not part of its training data. In other words, we solve this membership inference problem by converting it into a binary classification problem. We also study mitigation strategies to defend the ML models against the attacks discussed in this work.

We evaluate our method on real world datasets: (1) CIFAR-10 and (2) UCI Adult (Census Income) using classification as the task performed by the target ML models built on these datasets.

I. Introduction:

Machine Learning is the electricity and foundation of modern technologies and plays significant role in growing web-based services because of its wide applications. It is provided as service by Amazon, Google, Microsoft and many more. These companies provide services like training API, where the user can upload data to the cloud and train the model (example: A classification model). Later, user can use these models using prediction API's and do prediction. Prediction output is vector of probabilities that assign probability to each class to classify the object. Example in the Cifar-dataset, it takes a picture of a Car and assigns probability to the classes to predict whether it is a car, truck, airplane, submarine, etcetera.

These training API's are good examples of black box models, where the training model stays on the cloud and the user has no information about the architecture or parameters of the model, just can get the prediction vector. The user cannot even download the model anyhow! The prediction outputs have no information of the model nor information on predictions of the intermediate steps. Such black box models are very useful. Many mobile application developers use such services to predict the responses of the new features. We don't have access to the training datasets of the training model, so when we make prediction, there is no interaction with the dataset of the machine learning model, we just get the output prediction vector.

But, the real question is, do these machine learning models tend to leak information about their training data? We have learnt about the tendency of the leakage through studying membership inference attack against the machine learning models. Our aim is to find out as an attacker assuming that attacker has some information of the distribution or access to some part of the dataset, whether it was the part of the model's training dataset or not. It is challenging as we don't have direct access of the model or the dataset.

Idea: It is assumed that we get some access of the black box setting, i.e. we get to query it. Using some dataset distribution or samples that we have, we query the model and it creates some prediction vector stating its probability to be in some class. Now, we need to classify these distributions of the prediction so that we can infer that the dataset was in training or not.

To achieve our aim, we have turned machine learning against itself by training an attack model that studies the behavior of the target model on the training dataset that it is compiled and the datasets that it hasn't faced. **Classification!**

Shadow models have been created that imitates the behavior of the target model and the we have complete information about the training datasets used to train the shadow models. We then train the attack model using the labels of the inputs and the outputs of the shadow models. The training dataset used for shadow model is disjoint from the training dataset of the target model.

We have demonstrated the tendency of the black box model to leak the information using two datasets: CIFAR 10 and UCI Adult (Census Income). And, have shown mitigation strategy to decrease the probability of the attack.

II. Overview:

Machine learning model tend to behave differently with the training data as compared to the dataset that it hasn't seen. This phenomenon is called overfitting where the accuracy on training dataset is higher compared to testing dataset. The objective is to construct an attacking model that can classify the membership of the dataset used to query the target model.

Attack model is collection of 'k' attack model, each designed for 'k' different classes. This simply increases the attack accuracy as target model generates distribution of probabilities. We have used supervised learning to design multiple shadow model and used its labeled inputs and outputs to train the attack model. Formal setting is as described.

Suppose $m_{target}()$ is a target model and has a disjoint training dataset as D_{target}^{train} and contains labeled records in format of $\{x^i, y^i\}_{target}$ where x^i is the input data and y^i is it's true label taken from k classes. The predicted output is a vector of probabilities of 'k' size with probability ranging [0,1]. Summation of these probabilities is 1.

Similarly, $m_{attack}()$ is an attack model that takes input x_{attack}, which is combination of labeled record and prediction vector that is of size 'k'. This model is a binary classifier that infers the membership and outputs, 'out' or 'in'.

Figure (1) shows the entire process. Here, a record {x, y} is used by the target model to predict a vector $\hat{y}$ = $m_{target}()$. We pass $\{y, \hat{y}\}_{target}$ to the attack model. The attack model computes the probability whether the $\{y, \hat{y}\}_{target}$ is in training set or testing set of $m_{target}()$.

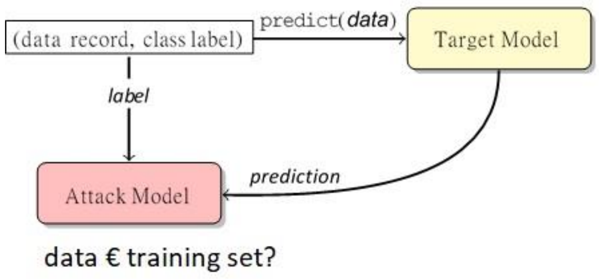


Figure 1: End-to-End Process

III. Shadow Models:

Setup: ‘n’ shadow models $m_{shadow}^{i}()$ are created by the attacker. Each i^{th} shadow model is trained on $D_{shadow^i}^{train}$, each of same type. In the worst case it is assumed that the $D_{shadow^i}^{train}$ and D_{target}^{train} may be disjoint. The shadow models are designed and trained in the same way as the target model (i.e. the user can use the same API used for training the target model if no information about the model architecture is known). As the number of shadow model increases, the accuracy of the attack increases. Figure (2) shows the above explanation.

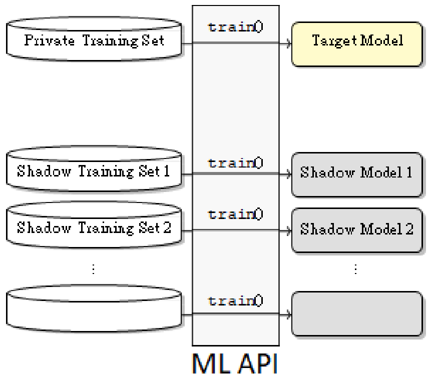


Figure 2: Shadow Model Trained using same API as the Target Model

IV. Attack Model:

The experimental setup of the training of the attack model is shown below. The setup shows that the shadow model’s output is used to train the attack model and it learns how to infer the membership of the dataset of the shadow model and thus produces a sequence to predict the membership of the training set of the target model.

The shadow model is queried using its own dataset for training and a disjoint testing dataset. The output generated by the training data is labelled as ‘in’ and the output by dataset for testing as ‘out’. This record is used to train the attack model.

Figure (3) shows, how we have trained the attack model. For each {x, y} € $D_{shadow^i}^{train}$, a prediction vector $\hat{y}$ along with its membership is added to the record of training set of attack model (y, $\hat{y}$, in). Similarly, for each {x, y} € $D_{shadow^i}^{test}$ we get a record (y, $\hat{y}$, out). A D_{attack}^{train} is formed using such records and is partitioned into ‘k’ partitions (k = number of classes). Each D_{attack}^{train} partitioned is associated to its respective class. It is like for each ‘y’ train a different model that would predict the membership for every x, given $\hat{y}$. Attack model is thus basically a binary classifier.

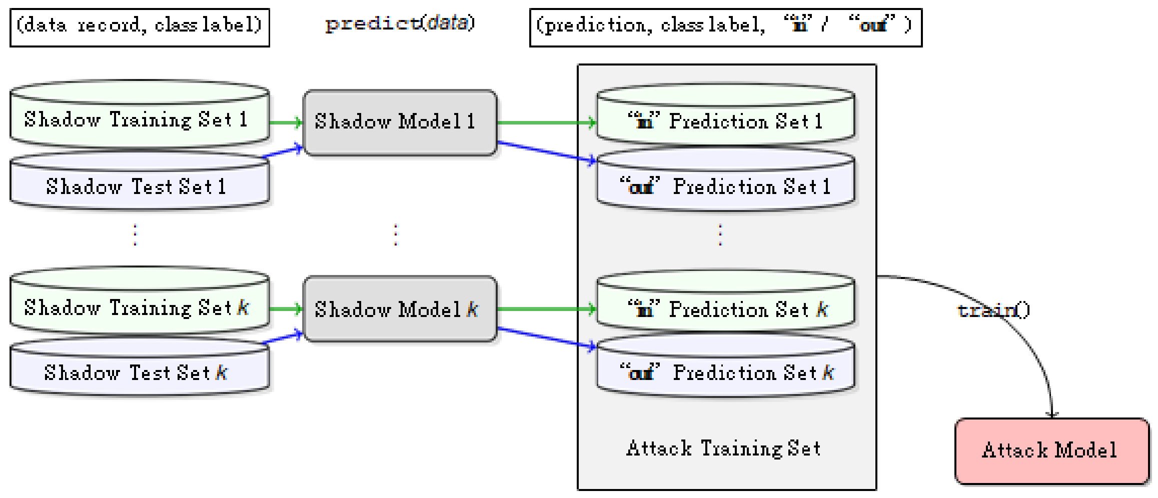


Figure 3: Training of Attack Model

V. Setup:

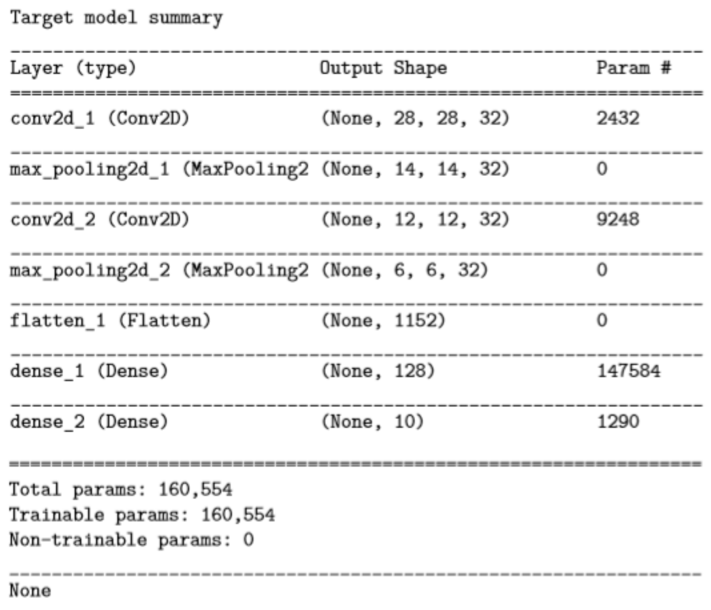
Description of the experimental setup for both CIFAR-10 and UCI Adult datasets is given below. The ***baseline accuracy*** for both experiments with shuffled dataset is assumed to be ***0.5***.

A) CIFAR:

CIFAR-10 is widely used in image recognition examples. It consists of 10 classes, each containing 6,000 32x32 color images per class. 50,000 for training and 10,000 for testing. Thus, in total there are 60,000 32x32 color images. We have combined both datasets and shuffled them all and then first extracted samples for target model and from the rest for the shadow model. By this we increase the probability for the dataset for target and shadow being disjoint. We have used distinct size of datasets in our experiment to picture the changes in the accuracy due to different datasize. The main aim for the classifier to is decide that the object belongs to which class.

Different datasize for our experiment are: 2500,5000,10000,15000 both for target and shadow model (i.e. 2500 each for training and testing of the target model and similarly for each shadow model). For each dataset, we have created 10 shadow models. The number of shadow models are selected according to number of classes for the datasets and here CIFAR 10 has 10 classes. We have compiled the target and shadow model using neural network. Two convolution layers with 'tanh' activation is used. The first convolution layer has kernel size = 5x5 and other has 3x3. Dense layer = 128 also with 'tanh' activation and at last a dense layer with 'softmax'. Categorical_entropy is used as a loss function because it is a multiclass classification. decay rate = 1e-7, learning rate = 0.001.

For attack model we have used SVM. To reach to the final result we have run a cross validation loop with nfold = 5 for C_test = [0.1,1,10] and gama_test = [0.001,0.01,0.1]. Through this we selected the best C and gamma for SVM.



```
Target model summary
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 28, 28, 32)        2432
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 32)          0
_________________________________________________________________
flatten_1 (Flatten)          (None, 1152)              0
_________________________________________________________________
dense_1 (Dense)              (None, 128)               147584
_________________________________________________________________
dense_2 (Dense)              (None, 10)                1290
=================================================================
Total params: 160,554
Trainable params: 160,554
Non-trainable params: 0
_________________________________________________________________
None
```

Figure 4: CIFAR10 Target Model Summary

For different datasize, we got average accuracy over total datasets as below:

For ds = 2500
Attack Precision: 0.7849293563579278
Attack Recall: 1.0
Attack Accuracy: 0.863

For ds = 5000
Attack Precision: 0.7275902211874272
Attack Recall: 1.0
Attack Accuracy: 0.8128

For ds = 10000
Attack Precision: 0.7211365111415591
Attack Recall: 1.0
Attack Accuracy: 0.80665

For ds = 15000
Attack Precision: 0.7089516967577276
Attack Recall: 1.0
Attack Accuracy: 0.7947333333333333

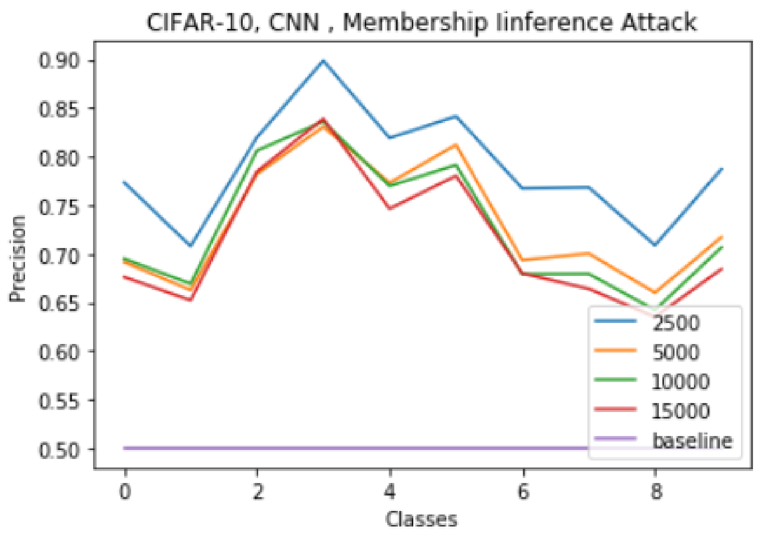
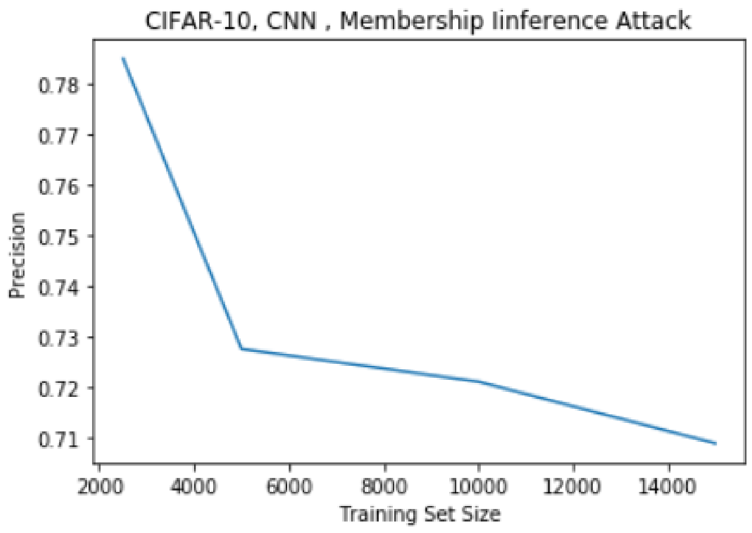


Figure 5: Precision v/s class Graph for different Datasize of CIFAR10

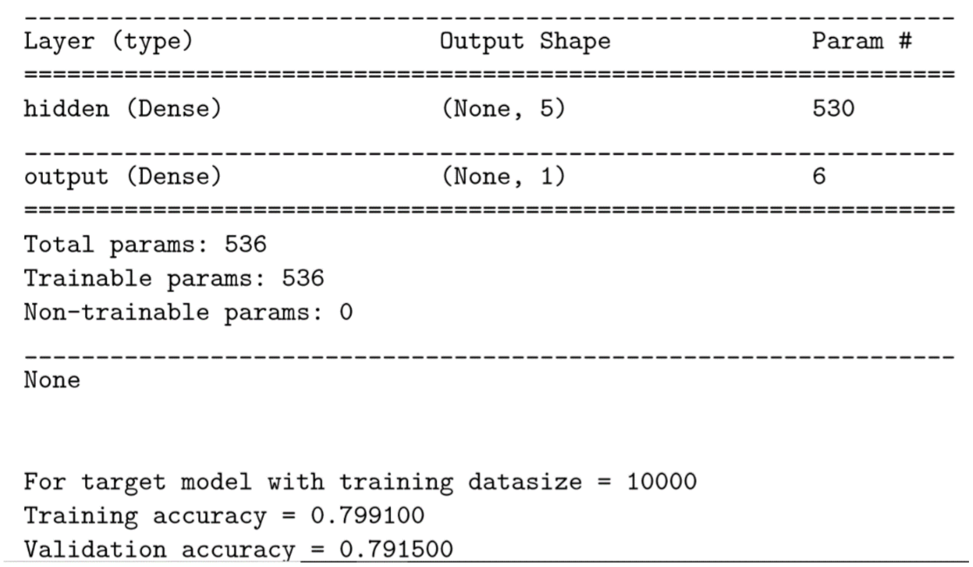


Observation: Our attack works. This is because our model has overfitted. Also, large number of classes makes the job of the model hard as it would have to go through lots of information. This lead, to leak of more information. We have given one mitigation strategy for that 'use of regularization' that helps to overcome overfitting and it works.

B) UCI Adult (Census Income):

This dataset has total 48,842 samples and 14 census features like gender, occupation, native country, marital status, age, working hours, education, race, etcetera. The main aim for the classifier is to predict whether a person earns more than $50K based on the features or not. Here we have randomly chose 10,000 train and testing samples for the target model and 10,000 random samples for training and testing shadow models.

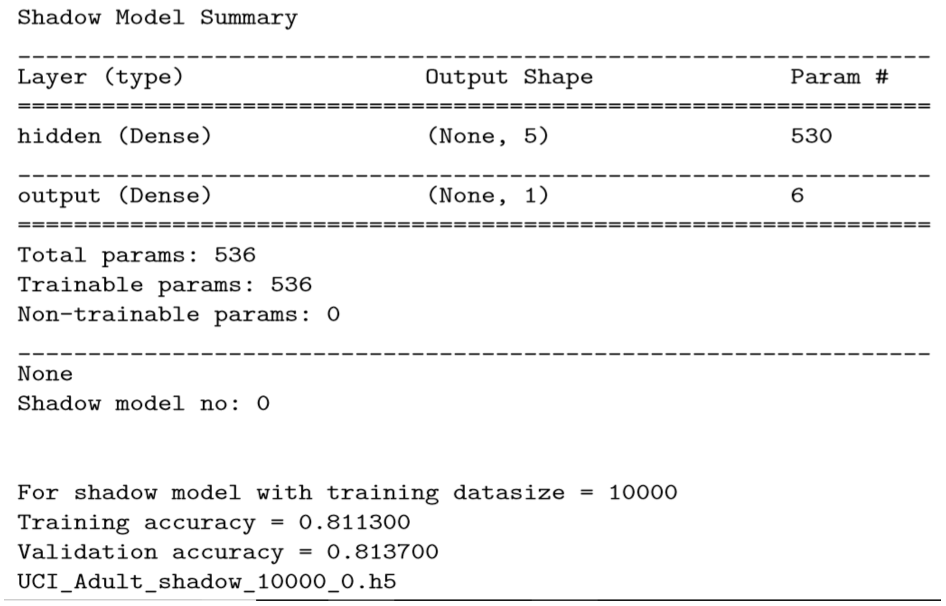
We have created 20 shadow models; each shadow model gets 10,000 training sample from shuffled and disjoint dataset than the one used for the target model. Number of shadow models can be increased to increase the prediction accuracy. We have compiled all the models on the *local machine*. We have used keras library with tensorflow working in the backend for creating neural networks. All the features with ‘object’ data types are one-hot-coded. Neural network with 5 hidden layers, decay rate = 1e-7, learning rate = 0.001 and sigmoid activation gives us 79.9% and 81% training accuracy for the target and shadow models respectively. 100 epochs are run for each model. Summary of the target and shadow model with its accuracy is shown below:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden (Dense)               (None, 5)                 530
_________________________________________________________________
output (Dense)               (None, 1)                 6
=================================================================
Total params: 536
Trainable params: 536
Non-trainable params: 0
_________________________________________________________________
None


For target model with training datasize = 10000
Training accuracy = 0.799100
Validation accuracy = 0.791500
```

Figure 6: UCI Adult Target Model Summary

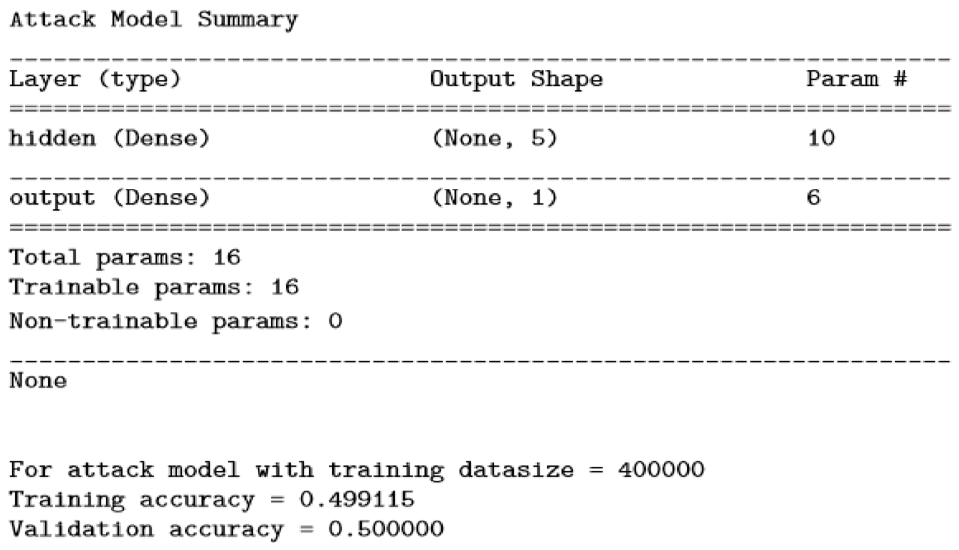


```
Shadow Model Summary
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden (Dense)               (None, 5)                 530
_________________________________________________________________
output (Dense)               (None, 1)                 6
=================================================================
Total params: 536
Trainable params: 536
Non-trainable params: 0
_________________________________________________________________
None
Shadow model no: 0


For shadow model with training datasize = 10000
Training accuracy = 0.811300
Validation accuracy = 0.813700
UCI_Adult_shadow_10000_0.h5
```

Figure 7: UCI Adult Shadow Model Summary

The output for the target and shadow model is a single class prediction output of probability whether the person earns more than $50K or not. ‘binary_crossentropy’ is used as the loss function. The same model format is used for the attack model as it is also a binary classifier. The attack model gets the training accuracy for 40,000 datasets to be 49.91 % and validation accuracy 50% (same as base line accuracy).



```
Attack Model Summary

_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
hidden (Dense)               (None, 5)                 10
_________________________________________________________________
output (Dense)               (None, 1)                 6
=================================================================
Total params: 16
Trainable params: 16
Non-trainable params: 0
_________________________________________________________________
None


For attack model with training datasize = 400000
Training accuracy = 0.499115
Validation accuracy = 0.500000
```

Figure 8: UCI Adult Attack model summary

There are two reasons why membership inference appeared to fail for this model. (1) As model is not overfitted, the training and testing accuracy are almost similar). (2) The model is a binary classifier as the attacker just must infer the membership by studying the behavior of the model with single class. Since the outputs are complimentary, it is not enough for the attack model to infer the membership information.

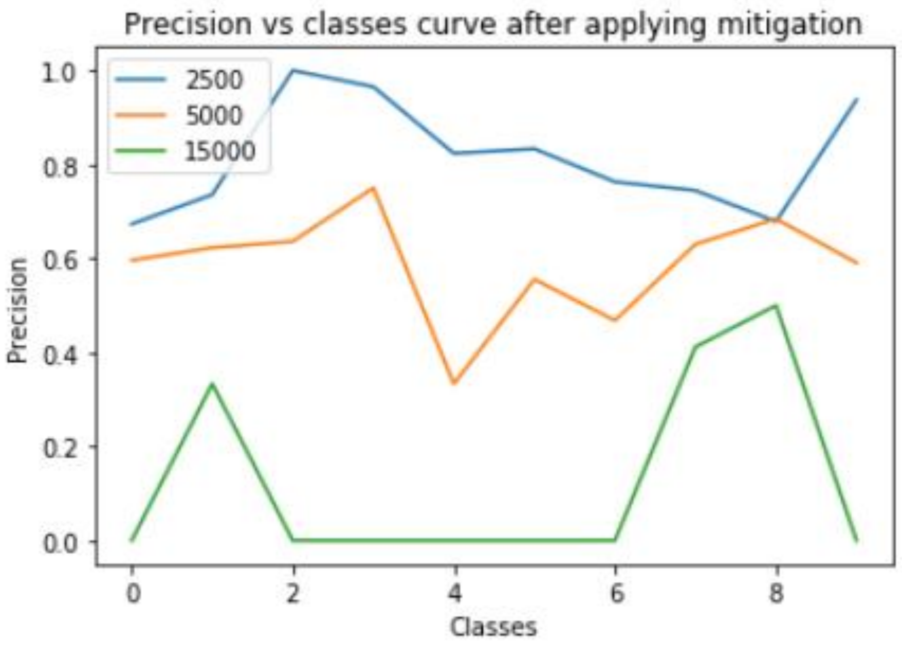
Observation: Models with few classes are less prone to leak their membership information. As the number of classes increase, the model gets more features from the sample to classify the input samples with higher accuracy. To simplify, models that have more classes have job to remember and classify more features and information about their training data and so are prone to leak more information.

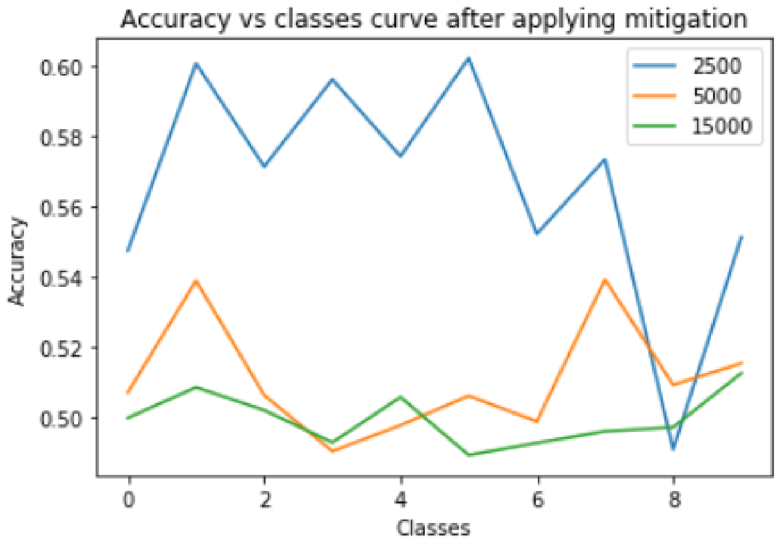
VI. <u>**Mitigation Strategy: (‘Use of Regularization)**</u>

Regularization techniques are normally used to overcome overfitting in machine learning. We have used L-2 regularization that penalizes large no of parameters. We have use lambda = 5e-3

For ds = 2500
Attack Precision: 0.7863247863247863
Attack Recall: 0.184
Attack Accuracy: 0.567
For ds = 5000
Attack Precision: 0.61
Attack Recall: 0.061
Attack Accuracy: 0.511

For ds = 15000
Attack Precision: 0.4074074074074074
Attack Recall: 0.0007333333333333333
Attack Accuracy: 0.49983333333333335


Precision vs classes curve after applying mitigation
Precision
Classes
2500
5000
15000
0.0
0.2
0.4
0.6
0.8
1.0
0
2
4
6
8


Accuracy vs classes curve after applying mitigation
Accuracy
Classes
2500
5000
15000
0.50
0.52
0.54
0.56
0.58
0.60
0
2
4
6
8