IE 506: Machine Learning - Principles and Techniques **Jan-May 2023**

Differentially Private Naïve Bayes Classification

Team Name: Sigma Enigma *Team Members: 22M0570 and 22M0574*

Abstract

There is increasing awareness of the need to protect individual privacy in the training data used to develop machine learning models. Differential Privacy is a strong concept of protecting individuals. Naïve Bayes is a popular machine learning algorithm, used as a baseline for many tasks. In this work, we have provided a differentially private Naïve Bayes classifier that adds noise proportional to the smooth sensitivity of its parameters. We compare our results to Vaidya, Shafiq, Basu and Hong [1] which scales noise to the global sensitivity of the parameters. Our experimental results on real-world datasets show that smooth sensitivity significantly improves accuracy while still guaranteeing ε-differential privacy.

1 Introduction

With the growth of user data across the internet, it has become more important to protect users' sensitive information. One solution to this problem is privacy- preserving data analysis, providing ability to share in- formation while protecting users' data. Dwork, McSherry, Nissim, and Smith [2] introduced differential privacy, providing a strong privacy guarantee for statistical data release. At a high level, Differential Privacy guarantees that the outcome of a differentially private algorithm would be similar no matter if a particular in- dividual contributes personal data to the database or not. There are several common approaches to differential privacy, including the Laplace Mechanism, which perturbs the parameters of the model with noise that is drawn from the Laplace distribution, scaled to the impact of a single individual on the result. The exponential Mechanism is another important mechanism to guarantee (ε, δ) differential privacy [3].

A model generated by a machine learning algorithm, when trained on a dataset, can reveal information about the training dataset. There are a series of recent works that guarantee that the output of a machine learning model satisfies differential privacy. These include differentially private Decision Trees [4], SVM [5], Deep Neural Networks [6], and Logistic Regression [7]. Differential privacy is particularly relevant for ensuring that ma- chine learning models do not disclose individual information and even has the promise of improving generalization [8]. Naïve Bayes is a baseline for many classification tasks. Vaidya, Shafiq, Basu, and Hong [1] provided a differentially private algorithm for the Naïve Bayes classifier. They use the Laplace Mechanism to provide this guarantee based on computing the global sensitivity of the parameters. One of the main drawbacks of using global sensitivity is that the amount of the noise added to the output can be high if there could be a dataset where an individual would have a large impact on an outcome. Nissim, Raskhodnikova, and Smith [9] provided a general solution for this problem. In their paper

they compute the *Smooth Sensitivity* for a given function f building on the definition of *local sensitivity*. The local sensitivity of f is the maximum amount of change in f if we change a single element in a *particular* dataset x. It is obvious that the local sensitivity of a given function is not greater than its global sensitivity. Ideally, we would like to add noise proportional to the local sensitivity of f, but this does not satisfy the definition of differential privacy (the amount of noise needed reveals too much about the data), hence, in [9] they compute a β smooth function which is the smallest upper bound for the local sensitivity that provides ε-differential privacy. In this paper, we show how this approach can be used to provide a ε-differentially private algorithm for the Naïve Bayes classifier based on the smooth sensitivity of the parameters of the model. Bun and Steinke [10] also provide an algorithm for estimating the mean of a distribution using i.i.d. sample x using smooth sensitivity. They first assume a crude bound on the $\mu \in [a, b]$, then they truncate the samples, i.e., they remove the largest m samples and smallest m samples from x, and compute the mean of the n-$2m$ samples. Finally, they project the estimated mean to the range $[a, b]$. We compare our algorithm to Vaidya, Shafiq, Basu, and Hong [1] and Naïve Bayes using Bun and Steinke [10]'s mean estimation algorithm.

2 Literature Survey

Dwork, McSherry, Nissim, Smith (2006) [2] proposed a detailed privacy definition of differential privacy. This paper was also one of the first to introduce the laplace mechanism through which the noise is added to the data, therefore preserving privacy in differential privacy. It also introduced the concept of sensitivity which measures how much a function's output change due to the change in one individual's data. Finally, they also showed that the scale of laplace distribution (with zero mean) depends on the value of global sensitivity and ε for a ε-differentially private mechanism. Dwork in [3] discusses about how a differentially private scheme can be achieved for a statistical database. It starts by introducing two settings, interactive and non-interactive, and how to secure privacy in these different environments. It then explains how noise can be added to put a veil on the original data and when the noise makes no sense. Finally, it also presents three differentially private algorithms and the general results about computational learning when privacy of individual is to be protected.

We have discussed the work of Vaidya et al. [1], which addresses the same problem we do using global sensitivity. Li et al. [17] proposed a new model for a differentially private Naïve Bayes classifier over multiple data sources. Their proposed method enables a trainer to train a Naïve Bayes classifier over the dataset pro- vided jointly by different data owners, without requiring a trusted aggregator as in our work and [1]. Yilmaz et al. [18] provided a differentially private Naïve Bayes classifier under the local differential privacy set- ting. With local differential privacy, individuals perturb their data before sending to an *untrusted* aggregator. The stronger adversary model of these two approaches (eliminating the trusted aggregator) results in significantly more noise and reduced accuracy.

3 Methods and Approaches

Dwork, McSherry, Nissim, and Smith [2] defined the notion of differential privacy. At a high level, differential privacy guarantees that if your data is a part of a database from which we release information, then the release information will be similar if your data is a part of the database or not. That is, your data will have a negligible impact on the released information. Hence, no meaningful information can be inferred about individuals.

One drawback of computing the global sensitivity of f is that for many functions, there may be some possible datasets, where changing one individual can make a dramatic change in the outcome. For example, suppose we are computing median on a value ranging from 0…1. The dataset consisting of individuals with values 0, 0, and 1 has median 0, but by changing one individual the median goes to 1 - so the added noise must essentially obscure the entire result. In practice, most databases will not have this property. Nissim, Raskhodnikova, and Smith [9] showed that we can add noise based on the actual dataset we have rather than a worst-case dataset, and still satisfy ε- differential privacy. We now outline their result.

The following definitions are useful in the application of this concept.

Definition 1: (Laplace Distribution): The probability density function (p.d.f) of the Laplace distribution $Lap(\mu, \lambda)$ is $\boldsymbol{f(x|\mu,\lambda)} = \frac{\mathbf{1}}{\mathbf{2\pi}}\boldsymbol{e^{-|x-\mu\}/\lambda}}$ with mean μ and standard deviation $\sqrt{2\lambda}$.

Definition 2: (Global Sensitivity): For f : D → R, the global sensitivity of f with respect to l_1 metric is:

$$\boldsymbol{GS_f} = \max_{\boldsymbol{x,y:d(x,y)=1}} ||\boldsymbol{f(x)} - \boldsymbol{f(y)}||_1$$

Definition 3: (Local Sensitivity): For f : D → R, the local sensitivity of f with respect to l_1 metric is:

$$\boldsymbol{LS_f} = \max_{\boldsymbol{y:d(x,y)=1}} ||\boldsymbol{f(x)} - \boldsymbol{f(y)}||_1$$

Method: The approach used to apply differential privacy in a naïve bayes classifier is by adding noise to the attribute data that is being used to calculate the final class label. Naïve Bayes classification involves calculating the likelihood and prior for the given dataset.

In order to add noise, a suitable distribution function is used. Laplace distribution has been found to give reasonably good results and it has been used in this work also. The parameters of laplace distribution depends on the sensitivity of the attribute data and is calculated separately for categorical and numeric data.

For numeric attributes, the sensitivity is taken as one since the inclusion or exclusion of one sample will change the count by one, and the probabilities depends on the no. of counts only. Whereas for numeric attributes, the sensitivity is found for the mean and standard deviation which are then used to calculate the probabilities.

The probability density function for a normal distribution with mean μ and variance σ^2 is given by

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

In order to add noise to the dataset, we need to decide the mean and scale of the laplace distribution.
The scale depends on the sensitivity and the privacy term Є which comes from the very definition of the differential privacy which states that – "A randomized algorithm is considered to be differentially private if for any pair of neighboring inputs, the probability of generating the same output, is within a small multiple of each other, for the entire output space. This means that for any two datasets which are close to one another, a differentially private algorithm will behave approximately the same on both data sets."

Mathematically, A randomized algorithm M with domain $\mathbb{N}^{|x|}$ is (Є, δ)- differentially private if for all S ∈ Range(M) and for all x, y ∈ $\mathbb{N}^{|x|}$ such that $||x-y||_1 \leq 1$:

$$\Pr[\mathcal{M}(x) \in \mathcal{S}] \leq \exp(\varepsilon) \Pr[\mathcal{M}(y) \in \mathcal{S}] + \delta,$$

In the current experiment, we've taken δ as zero.
The sensitivity in turn depends on the type of attribute, i.e., categorical or numeric.
For categorical, its equal to one whereas for numeric we have to calculate the sensitivity for the mean and standard deviation of the data according to the formula

For mean, Sensitivity = $\frac{(u_i - l_i)}{n+1}$

For standard deviation, Sensitivity = $\sqrt{n} \times \frac{(u_i - l_i)}{n+1}$

where u_j and l_j are the upper and lower bounds of the neighboring dataset and n is the no. of samples.

While the above description is theoretically complete, there are some things need to be considered during implementation. Specifically, when Laplacian noise is added, since the noise can be negative as well, it is possible for the mean, standard deviation, counts, and class priors to become negative. While this is not a problem for the mean, it is a problem for all of the rest. We've taken the absolute of the values to ensure that the modified values are non-negative.

As per our modification, the local sensitivity is also considered along with the global sensitivity, which involves calculating the sensitivity at every sample instead of calculating it for the worst case in the whole dataset. Since the no. of counts or the mean and standard deviation of the attributes change with every sample, while adding the noise we changed the mean from zero in global sensitivity to the no. of counts in local to add in the effect of the sample count.

3.1 Work done before mid-term project review

- Understanding the basic concepts of differential privacy and the Naïve bayes classification and the literature review of major works done in the last two decades was done along with a thorough study of the paper. First, naïve bayes theorem which is the root of the classification algorithm was studied along with appropriate assumptions that are made in making the formulation easier.
- The mathematical background involving the topic, like formulas, theorems and properties on differential privacy, naïve bayes theorem and laplace distribution that are used in implementing the algorithm were also studied.
- Basic outline of the code based on the algorithm given in the paper was discussed which was then worked upon after the mid-term.
- A modification to the current work was also proposed. The global sensitivity used in the study would be replaced by local sensitivity which results in smoothening of the results.

3.2 Work done after mid-term project review

- The algorithm for the differential private naïve bayes classification is implemented in a python code without using any predefined library for Differential privacy. The Naïve Bayes Classification model is used for the predicted values which requires prior and likelihood computations. These probabilities are found differently for different types of attributes viz. Categorical and Numeric and the code on itself gathers the data on the attribute type and run the model accordingly.
- The code is run for different datasets used in the original paper and the comparison is made between the two which shows that the results has been replicated well and the model is correct. A comparison is also made between the results obtained with and without the modification and all these are shown with the help of accuracy plots. Further details about the datasets are given in the next section.
- There are two important parameters which decide the level of privacy that is being incorporated in the model. While one of them is taken as zero ($\delta = 0$) which in fact increases the accuracy, the other term ϵ is of great importance as it helps us in finding the perfect trade-off between accuracy and privacy. The accuracy of the model after adding the noise is discussed and the variation of the accuracy for different values of the privacy parameter ϵ is also noticed. This helps us in finding an optimum value of ϵ.

4 Data set Details

Two datasets have been used in this project, one of which contains only categorical attributes while the other contains a combination of categorical and numeric.

1. Adult Dataset:
   Data Set Characteristics: Multivariate
   Number of Instances: 48842
   Attribute Characteristics: Categorical, Integer
   Number of Attributes: 14
2. Nursery Dataset:
   Data Set Characteristics: Multivariate
   Number of Instances: 12960
   Attribute Characteristics: Categorical
   Number of Attributes: 8

Only these two datasets have been taken since they span all the types of attributes possible and also contains a large enough samples and attributes so getting good accuracy on both these datasets will ensure good results on any dataset.

5 Experiments

Give details on experiments performed. Training procedure and algorithm, the settings used for optimization algorithm and other relevant algorithmic details need to be included. Details on the hardware configuration should also be described.

- **Algorithm:**

Algorithm 1 Computing differentially private parameters for Naïve Bayes

Require: ϵ, the privacy parameter for differential privacy
Require: Laplace(a, b) samples the Laplace distribution with mean a and scale b

for each attribute X_j **do**
  if X_j is categorical **then**
    sensitivity, $s \leftarrow 1$
    scale factor, $sf \leftarrow s/\epsilon$
    $\forall$ counts n_{kj}, $n'_{kj} = n_{kj}$+ Laplace$(0, sf)$
    Use n'_{kj} to compute $P(x_i|c_j)$
  else if X_j is numeric **then**
    compute sensitivity, s for mean μ_j as per equation 5
    scale factor, $sf \leftarrow s/\epsilon$
    $\mu'_j \leftarrow \mu_j$+ Laplace$(0, sf)$
    compute sensitivity, s for standard deviation σ_j as per equation 7
    scale factor, $sf \leftarrow s/\epsilon$
    $\sigma'_j \leftarrow \sigma_j$ + Laplace$(0, sf)$
    Use μ'_j and σ'_j to compute $P(x_i|c_j)$
  end if
end for
for each class c_j **do**
  count $nc'_j \leftarrow nc_j$+ Laplace$(0, 1)$
  Use nc'_j to compute the prior $P(c_j)$
end for

- **Script (Without Modification):**

```
import numpy as np
import pandas as pd
from scipy.stats import truncnorm

df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/nursery/nursery.data"

train_df = df.sample(frac=0.8,random_state=200)
test_df = df.drop(train_df.index)
train_y = train_df['Class']
train_X = train_df.drop('Class',axis=1)
test_y = test_df['Class']
test_X = test_df.drop('Class',axis=1)

cols = train_X.columns
num_cols = train_X._get_numeric_data().columns
num_count = len(num_cols)
cat_cols = list(set(cols) - set(num_cols))
cat_count = len(cat_cols)
column_names = list(df.columns)

unique_values_arr = []
for i in column_names :
  unique_values = df[i].unique()
  unique_values_arr.append(unique_values)

unique_values_arr_y = unique_values_arr[column_names.index('Class')]
unique_values_arr_X = np.delete(unique_values_arr,column_names.index('Class'),axis=0)
column_names_X = list(train_X.columns)

max_unique_values = 0
for i in column_names_X :
  unique_values = df[i].unique()
  if len(unique_values) >= max_unique_values  :
    max_unique_values  = len(unique_values)

grouped = train_df.groupby('Class')
count_Class =  train_df['Class'].value_counts()
order = unique_values_arr_y
count_y = count_Class.reindex(order)
```

```
def Accuracy_NB(epsilon):
  prob_cond_num = [[[0.0 for _ in range(len(unique_values_arr_y))] for _ in range(max_unique_values )] for _ in range(len(column_names_X))]
  for col in column_names_X:
    if df[col].dtype == 'object':
      s = 1
      sf = s/epsilon
      noise = abs(np.random.laplace(0,sf))
      for col in column_names_X:
        if train_df[col].dtype == 'object':
          for i in range(len(unique_values_arr_X[column_names_X.index(col)])):
            for k in range(len(unique_values_arr_y)):
              count = len(train_df[(train_df[col] == unique_values_arr_X[column_names_X.index(col)][i]) & (train_df['Class' ] == unique_values_arr_y[k])])  + noise
              prob = count/(count_y[unique_values_arr_y[k]]+noise)
              prob_cond_num[column_names_X.index(col)][i][np.where(unique_values_arr_y == unique_values_arr_y[k])[0][0]] = prob
    else:
      for i in unique_values_arr_y :
        df_class = grouped.get_group(i).drop('Class',axis=1)
        u = df_class[col].max()
        l = df_class[col].min()
        n = len(df_class)
        mean = df_class[col].mean()
        sensitivity_mean = (u-l)/(n+1)
        sf_mean = sensitivity_mean/epsilon
        mean_1 = mean + np.random.laplace(0,sf_mean)
        std = df_class[col].std()
        sensitivity_var = n**0.5* (u-l)/(n+1)
        sf_std = sensitivity_var/epsilon
        std_1 = std + np.random.laplace(0,sf_std)

        for j in range(len(unique_values_arr_X[column_names_X.index(col)])):
          x = unique_values_arr_X[column_names_X.index(col)][j]
          prob= 1/np.sqrt(2*np.pi)/std_1*np.exp(-((x-mean_1)**2)/2/std_1**2)
          prob_cond_num[column_names.index(col)][j][np.where(unique_values_arr_y == i)[0][0]] = prob

  Prior_prob = []
  noise_1 = abs(np.random.laplace(0,1))
  for Class in unique_values_arr_y:
    count_y_1 = count_y[Class] + noise_1
    prior = (count_y_1+1)/(len(train_df)+1)
    Prior_prob.append(prior)
```

- **Script (With Modification):**

```
import numpy as np
import pandas as pd
from scipy.stats import truncnorm
# np.random.seed(200)
```

```
df = pd.read_csv("https://archive.ics.uci.edu/ml/machine-learning-databases/nursery/nursery.data",names=["parents","has_nurs","form","chi
```

```
train_df = df.sample(frac=0.8,random_state=200)
test_df = df.drop(train_df.index)
```

```
train_y = train_df['Class']
train_X = train_df.drop('Class',axis=1)
test_y = test_df['Class']
test_X = test_df.drop('Class',axis=1)
```

```
cols = train_X.columns
num_cols = train_X._get_numeric_data().columns
num_count = len(num_cols)
cat_cols = list(set(cols) - set(num_cols))
cat_count = len(cat_cols)
print("number of numerical attributes of Dataset = ",num_count ,"\nnumber of categorical attributes of Dataset = ",cat_count)
```

```
number of numerical attributes of Dataset =  0
number of categorical attributes of Dataset =  8
```

```
column_names = list(df.columns)

unique_values_arr = []
for i in column_names :
  unique_values = df[i].unique()
  unique_values_arr.append(unique_values)
import warnings
warnings.filterwarnings("ignore")

unique_values_arr_y = unique_values_arr[column_names.index('Class')]
unique_values_arr_X = np.delete(unique_values_arr,column_names.index('Class'),axis=0)
```

```
column_names_X = list(train_X.columns)
```

```
max_unique_values = 0
for i in column_names_X :
  unique_values = df[i].unique()
  if len(unique_values) >= max_unique_values  :
    max_unique_values  = len(unique_values)
```

```
grouped = train_df.groupby('Class')
```

```
count_Class =  train_df['Class'].value_counts()
```

```
order = unique_values_arr_y
count_y = count_Class.reindex(order)
```

```
def Accuracy_NB(epsilon):
  prob_cond_num = [[[0.0 for _ in range(len(unique_values_arr_y))] for _ in range(max_unique_values )] for _ in range(len(column_names_X)
  for col in column_names_X:
    if df[col].dtype == 'object':
      s = 1
      for col in column_names_X:
        if train_df[col].dtype == 'object':
          for i in range(len(unique_values_arr_X[column_names_X.index(col)])):
            for k in range(len(unique_values_arr_y)):
              count = len(train_df[(train_df[col] == unique_values_arr_X[column_names_X.index(col)][i]) & (train_df['Class'] == unique_va
              count = np.random.laplace(count, s/epsilon)
              count = max(count,0)
              prob = count/(count_y[unique_values_arr_y[k]])
              prob_cond_num[column_names_X.index(col)][i][np.where(unique_values_arr_y == unique_values_arr_y[k])[0][0]] = prob
    else:
      for i in unique_values_arr_y :
        df_class = grouped.get_group(i).drop('Class',axis=1)

        u = df_class[col].max()
        l = df_class[col].min()
        n = len(df_class)
```

```
        mean = df_class[col].mean()
        sensitivity_mean = (u-l)/(n+1)
        sf_mean = sensitivity_mean/epsilon
        std = df_class[col].std()
        sensitivity_var = n**0.5* (u-l)/(n+1)
        sf_std = sensitivity_var/epsilon
        for j in range(len(unique_values_arr_X[column_names_X.index(col)])):
          x_1 = unique_values_arr_X[column_names_X.index(col)][j]
          x = np.random.laplace(mean,sf_mean)+np.random.laplace(std,sf_std)
          prob= 1/np.sqrt(2*np.pi)/std*np.exp(-((x-mean)**2)/2/std**2)
          prob_cond_num[column_names.index(col)][j][np.where(unique_values_arr_y == i)[0][0]] = prob
  Prior_prob = []
  noise_1 = abs(np.random.laplace(0,1))
  for Class in unique_values_arr_y:
    count_y_1 = count_y[Class] + noise_1
    prior = (count_y_1+1)/(len(train_df)+1)
    Prior_prob.append(prior)

  def Probability_NB(sample_no):
    prob_NB = []
    for i in range(len(unique_values_arr_y)) :
      prob = 1
      test_class = test_X.iloc[sample_no]
      A = test_class.to_numpy()
      att = []
      for k in range(len(test_class)):
          att.append(test_class[k])
          probability_att = prob_cond_num[k][np.where(unique_values_arr_X[k] == A[k])[0][0]][i]
          prob = probability_att*prob
      prob_NB.append(prob*Prior_prob[i])
    return prob_NB.index(max(prob_NB))

  test_y_arr = test_y.to_numpy()
  Count_p = 0
  for i in range(len(test_y)):
    if unique_values_arr_y[Probability_NB(i)] == test_y_arr[i]:
      Count_p += 1
  Accuracy = Count_p/len(test_y)*100
  return Accuracy
```

```
Epsilon = [0.01,0.1,1]
```

```
Accuracy_arr = [0.0  for _ in range(len(Epsilon))]
for i in range(len(Epsilon)):
  Accuracy_arr[i] = Accuracy_NB(Epsilon[i])

print(Accuracy_arr)
```

```
[88.04012345679013, 91.5895061728395, 88.00154320987654]
```

```
import matplotlib.pyplot as plt
plt.gca().invert_xaxis()
plt.xscale("log")
plt.plot(Epsilon, Accuracy_arr, 'r-')
plt.legend()
plt.xlabel('Epsilon')
plt.ylabel('Accuracy')
plt.title('Nursery dataset: 13K records, 8 attributes and 5 classes')
plt.show()
```

6 Results

- Plot

Nursery dataset:

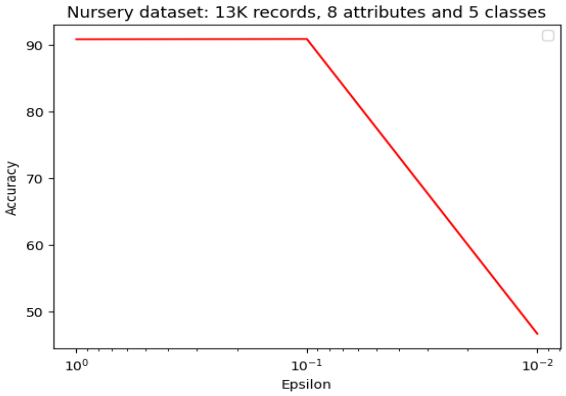

Fig. Accuracy vs Epsilon (Without modification)

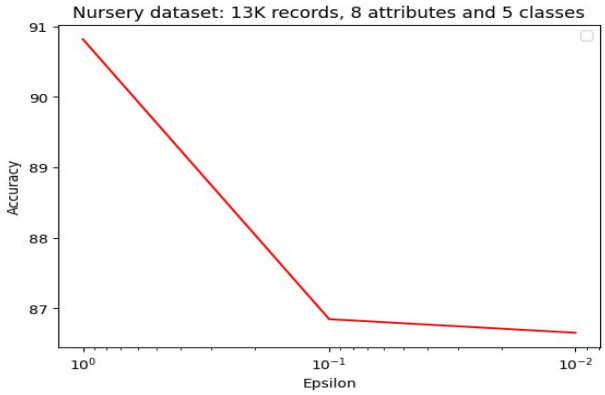

Fig. Accuracy vs Epsilon (With modification)

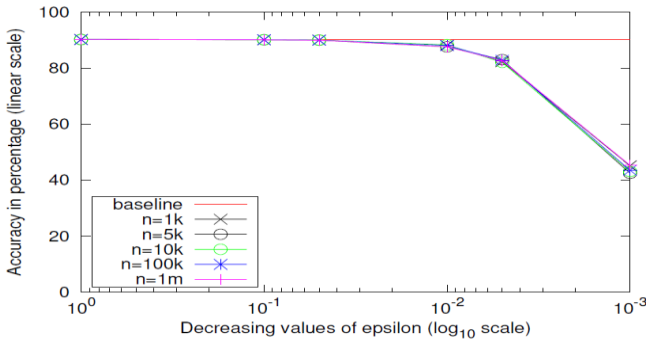

Fig. Accuracy vs Epsilon (Vaidya et al., 2016)

Adult dataset:

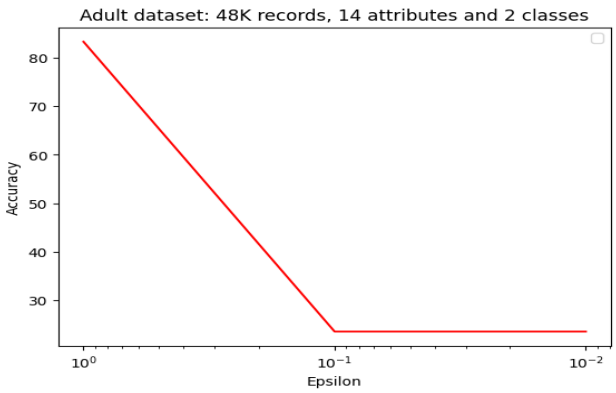

Fig. Accuracy vs Epsilon (Without modification)

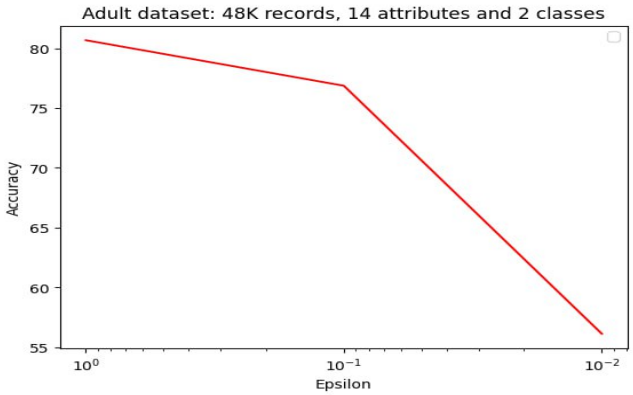

Fig. Accuracy vs Epsilon (With modification)

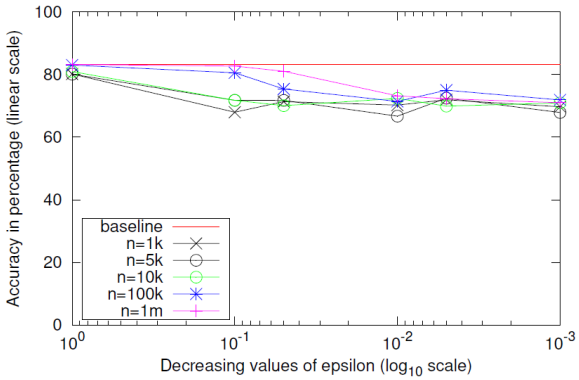

Fig. Accuracy vs Epsilon (Vaidya et al., 2016)

7 Future Work

- Instead of Naïve Bayes, other classification techniques can also be made differentially private.
- Regression models can also be provided better privacy as in the case of continuous class, classification models do not work and regression analysis is used.
- Other privacy-preserving techniques such as secure multi-party computation and homomorphic encryption can also be used.
- The model can be made to work on even bigger datasets with millions of samples and attributes.

8 Conclusion

- We have successfully implemented the code using the algorithm provided in the paper and have tested the model for different databases with varying no of samples, attributes and classes.
- The accuracy in each case is calculated and found satisfactory in the range of 20 to 90% for adult dataset (for different values of Є) and 50-90% for the nursery dataset without modification while 90% for adult dataset (for different values of Є) and 80-90% for the nursery dataset with modification.
- Three different values of the parameter Є has been chosen for determining the variation of accuracy and it has been found that the accuracy increases with increment of Є which confirms the trend as seen in the paper.
- The code also runs with satisfactory result for datasets with mixed type of attributes i.e., both categorical and numeric.
- Due to the random nature of the noise added to the attributes, the results changes with every run of the code but the value remains satisfactory.

References

- J. Vaidya, B. Shafiq, A. Basu, and Y. Hong, "Differentially private naive bayes classification," in 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), vol. 1, pp. 571– 576, IEEE, 2013.
- C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in TCC, 2006, pp. 265–284.
- Dwork, C. (2008). Differential Privacy: A Survey of Results. In: Agrawal, M., Du, D., Duan, Z., Li, A. (eds) Theory and Applications of Models of Computation. TAMC 2008. Lecture Notes in Computer Science, vol 4978. Springer, Berlin, Heidelberg.
- J. Vaidya, C. Clifton, and M. Zhu, Privacy-Preserving Data Mining, 1st ed., ser. Advances in Information Security. Springer-Verlag, 2005, vol. 19.
- Cynthia Dwork; Aaron Roth, The Algorithmic Foundations of Differential Privacy , now, 2014.