presentation

September 16, 2022

0.1 Data Privacy and Anonymization

[1]:
```
import numpy as np
import pandas as pd
from pandas.api.types import CategoricalDtype
from IPython.display import HTML
from sklearn.model_selection import train_test_split
from sklearn.decomposition import PCA
from diffprivlib.models import RandomForestClassifier
import diffprivlib as dp
from sklearn.multioutput import MultiOutputClassifier
from sklearn.metrics import f1_score, classification_report
from IPython.display import Image
from IPython.core.display import HTML
```

[2]:
```
# specify all the datatypes of the dataset
dtypes = {
    "County": object,
    "Gender": CategoricalDtype(),
    "Martial_status": "category",
    "No_of_dependents": np.int32,
    "Age": np.int32,
    "Tier": CategoricalDtype(categories=["Bronze", "Silver", "Gold"],
                             ordered=True),
    "Policy_limit": np.float32,
    "Illness_category": "category",
    "Updated_subscription": object,
    "Updated_subscription": object,
    "Account_withdrawal": object,
    "Retention": np.int32,
    "ID_number": np.int32,
    "Tax_id": object
}

dtypes2 = {
    "Instrument": object,
    "Reading": object,
```

```
    "Observation": np.float32,
    "Status": "category"
}
```

[3]:
```
# load all the datasets and specify the arguments to making parsing easier
insurance_df = pd.read_csv("Insurance_data_ke.csv",
                           index_col="id",
                           parse_dates=["Date_of_entry"],
                           dtype=dtypes)
anonymous_df = pd.read_csv("Insurance_data_ke.csv", header=None, skiprows=1)
spo2_temp_hr_df = pd.read_csv("Organs.csv",
                              parse_dates=["Date"],
                              dtype=dtypes2)
```

Thank the audience for coming and I will go through it in a breeze. TGIF activities and joining them.

What I'll talk about: * Definitions * Techniques: * Data Deletion! * Masking & Sampling * Privacy Models: K-anonymity, PCA and Differential privacy + ML

0.2 Data Privacy

Is the protection of personal data from those who should not have access to it and the ability of individuals to determine who can access their personal information.

Authorized, fair, and legitimate processing of personal information. Credit: *Michelle Dennedy, Jonathan Fox, Tom Finneran, The Privacy Engineer's Manifesto (Apress, 2014), p. 34.*

Personally identifiable information (PIIs)

Sensitive PIIs - info that can be directly liked to a person e.g **biometric data**, **genetic data**, **health status**, **National ID number**, **Passport number**, **KRA pin**

Non-sensitive PIIs- info can't be directly linked to a person as an example **birthday**, **county of origin**

Quasi-identifying - on their own they are not identifying but when combined can be. **Latanya Sweeney** confirmed that combining birth dates, gender and postcodes can be used to identify people in the United States. She also discovered something interesting with her name. Find out. What about Kenya?

Anonymization

Removing PIIs in datasets to keep the individuals Jane/John Does.

0.3 Data Deletion

Plan for it:
* Come up with a data inventory and control access
* Follow that up with a segregation strategy: Operational data (10 months) and archival data

(5 years) * Create a Retriever: Gets all the data regarding a specific client. It's their right. * Create a Destroyer: Deletes all data regarding a customer from every store e.g database, object storage(object expiration) en cetera * Make a privacy engineering team to handle these.

from Imgflip Meme Generator

Enter the matrix

from Imgflip Meme Generator

0.4 Compare Datasets

[4]:
```
# a summary of the data
anonymous_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1001 entries, 0 to 1000
Data columns (total 15 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   0       1001 non-null   int64
 1   1       1001 non-null   object
 2   2       1001 non-null   object
 3   3       1001 non-null   object
 4   4       1001 non-null   int64
 5   5       1001 non-null   int64
 6   6       1001 non-null   object
 7   7       1001 non-null   int64
 8   8       1001 non-null   object
 9   9       1001 non-null   object
 10  10      1001 non-null   object
 11  11      1001 non-null   object
 12  12      1001 non-null   int64
 13  13      1001 non-null   int64
 14  14      1001 non-null   object
dtypes: int64(6), object(9)
memory usage: 117.4+ KB
```

0.5 Masking

[6]:
```
# Masking the data
anonymous_df.head()
```

[6]:
```
   0         1       2         3  4   5       6       7          8  \
0  0  Machakos  Female   Married  2  50  Bronze  300000  Coegnital
1  1     Narok  Female    Single  0  52    Gold  300000  Coegnital
2  2     Kwale  Female  Divorced  2  40  Bronze  500000   subacute
```

```
3   3  Machakos    Male    Married   4  34  Bronze  1000000      acute
4   4      Busia  Female   Married   4  36    Gold  1000000   subacute

                     9   10  11  12        13           14
0  2017-04-17 18:50:16   No  No  18  23786861  D092653961D
1  2016-05-25 11:35:11  Yes  No  17  35088212  A912594412N
2  2012-05-17 07:23:16  Yes  No  21   8851699  X565831552H
3  2015-02-21 04:59:58   No  No  15   1977748  W009533562Y
4  2013-07-27 06:06:43   No  No  18  23978004  C311215825F
```

[7]:
```
insurance_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1001 entries, 0 to 1000
Data columns (total 14 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   County                 1001 non-null   object
 1   Gender                 1001 non-null   category
 2   Martial_status         1001 non-null   category
 3   No_of_dependents       1001 non-null   int32
 4   Age                    1001 non-null   int32
 5   Tier                   1001 non-null   category
 6   Policy_limit           1001 non-null   float32
 7   Illness_category       1001 non-null   category
 8   Date_of_entry          1001 non-null   datetime64[ns]
 9   Updated_subscription   1001 non-null   object
 10  Account_withdrawal     1001 non-null   object
 11  Retention              1001 non-null   int32
 12  ID_number              1001 non-null   int32
 13  Tax_id                 1001 non-null   object
dtypes: category(4), datetime64[ns](1), float32(1), int32(4), object(4)
memory usage: 70.9+ KB
```

0.6 Switch to HTML report

1 HTML(filename='insurance_report.html')

[8]:
```
# remove columns that are very particular
insurance_df = insurance_df.drop(["ID_number", "Tax_id"], axis=1)
```

[9]:
```
# see what columns were left
insurance_df.columns
```

```
[9]: Index(['County', 'Gender', 'Martial_status', 'No_of_dependents', 'Age', 'Tier',
       'Policy_limit', 'Illness_category', 'Date_of_entry',
       'Updated_subscription', 'Account_withdrawal', 'Retention'],
      dtype='object')
```

1.1 Sampling

```
[10]: # make a subset of the dataframe
insurance_df_sample = insurance_df.sample(n=100)
```

```
[11]: # finding unique iteams in the pandas Series and return a probability
insurance_df['Martial_status'].value_counts(normalize=True)
```

```
[11]: Married     0.602398
Single      0.216783
Divorced    0.180819
Name: Martial_status, dtype: float64
```

```
[12]: mar_status_counts = insurance_df_sample['Martial_status'].value_counts(
    normalize=True)

mar_status_counts
```

```
[12]: Married     0.56
Single      0.23
Divorced    0.21
Name: Martial_status, dtype: float64
```

```
[13]: # make a non random sample distribution based on the original dataset
insurance_df_sample["Martial_status"] = np.random.choice(
    mar_status_counts.index,
    p=mar_status_counts.values,
    size=len(insurance_df_sample))
```

```
[14]: insurance_df_sample['Martial_status'].value_counts(normalize=True)
```

```
[14]: Married     0.57
Divorced    0.24
Single      0.19
Name: Martial_status, dtype: float64
```

1.2 K-anonymity

Reducing distinction between groups. K groups share properties.

from Imgflip Meme Generator

[15]:
```
# arrange dataframe by Age and tier, count instances of both and rename the␣
↪column
insurance_df.groupby(["Age", "Tier"]).size().reset_index(name="Count")
```

[15]:
```
     Age    Tier  Count
0     18  Bronze     12
1     18  Silver      4
2     18    Gold      1
3     19  Bronze      8
4     19  Silver      2
..   …      …     …
166   73  Silver      7
167   73    Gold      4
168   74  Bronze     11
169   74  Silver      1
170   74    Gold      3

[171 rows x 3 columns]
```

[16]:
```
# binning data into at most 4 intervals
insurance_df["Age"] = pd.cut(insurance_df["Age"], bins=4)
```

[17]:
```
# see sample
insurance_df[["Age", "Tier"]]
```

[17]:
```
                  Age    Tier
id
0        (46.0, 60.0]  Bronze
1        (46.0, 60.0]    Gold
2        (32.0, 46.0]  Bronze
3        (32.0, 46.0]  Bronze
4        (32.0, 46.0]    Gold
…                 …     …
996    (17.944, 32.0]  Silver
997      (60.0, 74.0]    Gold
998      (60.0, 74.0]  Bronze
999      (46.0, 60.0]  Bronze
1000     (46.0, 60.0]  Bronze

[1001 rows x 2 columns]
```

[18]:
```
# redistributes the information
insurance_df["Age"].value_counts(normalize=True)
```

[18]:
```
(17.944, 32.0]    0.267732
(46.0, 60.0]      0.252747
(32.0, 46.0]      0.239760
(60.0, 74.0]      0.239760
Name: Age, dtype: float64
```

[19]:
```
# How many specific groups arise should not be 4
k = 4

count_groups = insurance_df.groupby(["Age",
                                     "Tier"]).size().reset_index(name="Count")

count_groups[count_groups['Count'] < k]
```

[19]:
```
Empty DataFrame
Columns: [Age, Tier, Count]
Index: []
```

1.3 Differential Privacy

Adding statistical noise to your work. Governed by an epsilon parameter which is the **privacy budget**. Low values give less accurate data whereas high values give the most accurate data. Do you want someone knowing that you have a short lifespan given your multimorbidity or what is the staple food in Kenya?

from Imgflip Meme Generator

[20]:
```
# make 100 random numbers off a normal distribution
np.random.seed(1)
X = np.random.randint(1, 10, size=10, dtype=np.int32)

# specify budget accountant, adjust value to be 10 and see the difference
acc = dp.BudgetAccountant(epsilon=1.0)

# find the mean
print(X)
print(f"Then mean of your array is",
      dp.tools.mean(X, bounds=(1, 9), accountant=acc),
      ",Using the normal mean function", {np.mean(X)})
print("Budget so far ", acc.remaining())  # how much you have spen
```

```
[6 9 6 1 1 2 8 7 3 5]
Then mean of your array is 4.839219885699528 ,Using the normal mean function
{4.8}
Budget so far  (epsilon=1.1102230246251565e-16, delta=1.0)
```

[20]:
```
# run it again, should not work
#print(f"Then mean of your array is",
#       {dp.tools.mean(X, bounds=(1, 8), accountant=acc)},
#       "Using the normal mean function", {np.mean(X)})
#print("Budget so far ", acc.remaining())
```

[21]:
```
# load dataset into memory
feat_eng_df = pd.read_csv("feature_engineered_insurance.csv")
```

[22]:
```
feat_eng_df.columns
```

[22]:
```
Index(['No_of_dependents', 'Age', 'Policy_limit', 'Retention',
       'Date_of_entryYear', 'Date_of_entryMonth', 'Date_of_entryWeek',
       'Date_of_entryDay', 'Date_of_entryDayofweek', 'Date_of_entryDayofyear',
       'Account_withdrawal=No', 'Account_withdrawal=Yes', 'County=Baringo',
       'County=Bomet', 'County=Bungoma', 'County=Busia',
       'County=Elgeyo-Marakwet', 'County=Embu', 'County=Garissa',
       'County=Homa Bay', 'County=Isiolo', 'County=Kajiado', 'County=Kakamega',
       'County=Kericho', 'County=Kiambu', 'County=Kilifi', 'County=Kirinyaga',
       'County=Kisii', 'County=Kisumu', 'County=Kitui', 'County=Kwale',
       'County=Laikipia', 'County=Lamu', 'County=Machakos', 'County=Makueni',
       'County=Mandera', 'County=Marsabit', 'County=Meru', 'County=Migori',
       'County=Mombasa (County)', 'County=Murang'a', 'County=Nairobi (County)',
       'County=Nakuru', 'County=Nandi', 'County=Narok', 'County=Nyamira',
       'County=Nyandarua', 'County=Nyeri', 'County=Samburu', 'County=Siaya',
       'County=Taita-Taveta', 'County=Tana River', 'County=Tharaka-Nithi',
       'County=Trans-Nzoia', 'County=Turkana', 'County=Uasin Gishu',
       'County=Vihiga', 'County=Wajir', 'County=West Pokot',
       'Date_of_entryIs_month_end', 'Date_of_entryIs_month_start',
       'Date_of_entryIs_quarter_end', 'Date_of_entryIs_quarter_start',
       'Date_of_entryIs_year_end', 'Date_of_entryIs_year_start',
       'Gender=Female', 'Gender=Male', 'Illness_category=Coegnital',
       'Illness_category=acute', 'Illness_category=chronic',
       'Illness_category=subacute', 'Martial_status=Divorced',
       'Martial_status=Married', 'Martial_status=Single', 'Tier=Bronze',
       'Tier=Gold', 'Tier=Silver', 'Updated_subscription=No',
       'Updated_subscription=Yes', 'id'],
      dtype='object')
```

[23]:
```
# preparing the matrix and the vector
# Dependent variable/predictors: X and independent variable/Target: If the␣
 ↪client is going to stopped using the service
feat_eng_df2 = feat_eng_df.drop(["id"], axis=1)
x = feat_eng_df2.to_numpy()
y = np.random.choice([0, 1], p=[0.50, 0.50],
                     size=len(feat_eng_df))  # pareto principle?
y
```

[23]: array([0, 1, 0, …, 0, 1, 1])

[24]:
```
# Review the sample of array
x.shape
```

[24]: (1001, 79)

[25]:
```
y.shape
```

[25]: (1001,)

[26]:
```
# model validation strategy: Train test split
features_train, features_validation_test, labels_train, labels_validation_test␣
↪= train_test_split(
    x, y, test_size=0.4, random_state=100)
features_validation, features_test, labels_validation, labels_test =␣
↪train_test_split(
    features_validation_test,
    labels_validation_test,
    test_size=0.5,
    random_state=100)
```

[27]:
```
# Specify, fit, Predict paradigm
clf = RandomForestClassifier(n_jobs=-1, epsilon=2, random_state=345, verbose=1)

clf.fit(features_train, labels_train)

preds = clf.predict(features_validation_test)

report = classification_report(labels_validation_test, preds)
print(report)
```

```
/home/bens/anaconda3/envs/data-privacy-env/lib/python3.9/site-
packages/diffprivlib/models/forest.py:188: PrivacyLeakWarning: `feature_domains`
parameter hasn't been specified, so falling back to determining domains from the
data.
This may result in additional privacy leakage. To ensure differential privacy
with no additional privacy loss, specify `feature_domains` according to the
documentation
  warnings.warn(
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 4 concurrent workers.

              precision    recall  f1-score   support

           0       0.50      0.60      0.55       198
           1       0.52      0.42      0.46       203

    accuracy                           0.51       401
```

```
   macro avg       0.51      0.51      0.51       401
weighted avg       0.51      0.51      0.50       401

[Parallel(n_jobs=-1)]: Done  10 out of  10 | elapsed:   58.9s finished
[Parallel(n_jobs=4)]: Using backend ThreadingBackend with 4 concurrent workers.
[Parallel(n_jobs=4)]: Done  10 out of  10 | elapsed:    0.1s finished
```

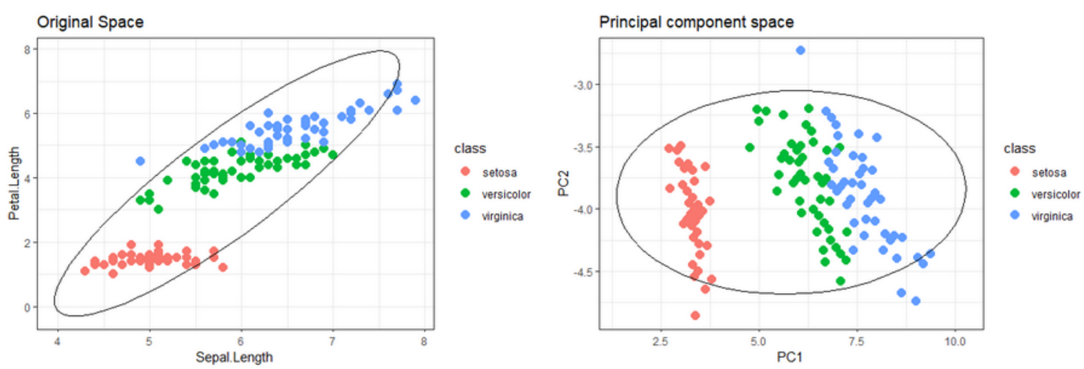
Use the newer version of diffprivlib to tune it better. Try running getting a model without using diffprivlib.

1.4 PCA

Principal component Analysis. Very pervasive technique to reduce the dimensions of any dataset and still retaining the properties of the data. You can also use it for data compression.

[32]:
```
PATH = "/home/bens/Desktop/data_privacy_004/"
Image(filename=PATH + "Screenshot from 2022-09-10 07-03-38.png",
      width=500,
      height=500)
```

[32]:



[33]:
```
def dim_reduction(data):
    pca = PCA(n_components=2)
    return pca.fit_transform(data)
```

[34]:
```
feat_eng_df2.memory_usage()
```

[34]:
```
Index                     128
No_of_dependents         8008
Age                      8008
Policy_limit             8008
Retention                8008
                          …
Tier=Bronze              8008
Tier=Gold                8008
```

```
Tier=Silver                   8008
Updated_subscription=No       8008
Updated_subscription=Yes      8008
Length: 80, dtype: int64
```

[35]:
```
# apply pca
pca_mat = dim_reduction(feat_eng_df2)
```

[36]:
```
# change the resultant matrix to a dataframe
pca_df = pd.DataFrame(pca_mat)
```

[37]:
```
pca_df.memory_usage()
```

[37]:
```
Index     128
0        8008
1        8008
dtype: int64
```

Moved from 80 columns to 2. Notice: the number of bytes in the column resembles that of the original column entries.

[38]:
```
# How would you scan this dataset with your current knowledge
spo2_temp_hr_df.info()
print("#" * 100)
spo2_temp_hr_df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 693 entries, 0 to 692
Data columns (total 5 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         693 non-null    object
 1   Instrument   693 non-null    object
 2   Reading      693 non-null    object
 3   Observation  693 non-null    float32
 4   Status       693 non-null    category
dtypes: category(1), float32(1), object(3)
memory usage: 20.0+ KB
################################################################################
####################
```

[38]:

	Date	Instrument	Reading	Observation	Status
0	2021/04/04 12:05:00	Pulse oximeter	HR	62.000000	Well
1	2021/04/04 12:06:00	Pulse oximeter	SpO2	96.000000	Well
2	2021/04/04 12:13:42	Thermometer	Temperature	36.700001	Well
3	2021/10/04 09:33:00	Thermometer	Temperature	37.000000	Well
4	2021/10/04 09:33:00	Pulse oximeter	HR	63.000000	Well

[39]:
```
spo2_temp_hr_df.Status.value_counts(normalize=True)
```

[39]:
```
Well       0.917749
Tired      0.056277
Allergy    0.012987
anxious    0.008658
Off        0.004329
Name: Status, dtype: float64
```

Summary:
Definitions:
* **Data Privacy**: segregating data and controlling access. "Gatekeeping data"
* **Anonymization**: removing PIIs
* PII information that can identify you either directly sensitive PIIs or indirect Quasi-identifiers. Non-sensitive PIIs are not dangerous but they are
Techniques:
* **Data Deletion** - is a way of reducing load on your object storage and database. Make a solid plan and talk to legal team to advice. A retriever and a destroyer is a good start. You even save money.
* **Masking & Sampling** - storing data without column names in for instance object storage and giving a small subset still retaining properties of the original dataset.
* **K-anonymity** - making groups that exist have similar characteristics reducing reidentification strategies. Look into l-diversity.
* **Differential privacy ML** - adding noise to your models governed by BudgetAccountant and epsilon parameter.

Learn more

- https://www.manning.com/books/data-privacy
- https://ethics.fast.ai/
- https://www.apple.com/privacy/docs/Differential_Privacy_Overview.pdf
- https://www.manning.com/books/privacy-preserving-machine-learning
- https://www.manning.com/books/grokking-deep-learning
- https://www.manning.com/books/build-a-career-in-data-science
- https://www.datacamp.com/