```
!pip install pyro-ppl
```

```
Collecting pyro-ppl
  Downloading pyro_ppl-1.9.0-py3-none-any.whl (745 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 745.2/745.2 kB 10.1 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.7 in /usr/local/lib/python3.10/dist-packages (from pyro-ppl) (1.25.2)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from pyro-ppl) (3.3.0)
Collecting pyro-api>=0.1.1 (from pyro-ppl)
  Downloading pyro_api-0.1.2-py3-none-any.whl (11 kB)
Requirement already satisfied: torch>=2.0 in /usr/local/lib/python3.10/dist-packages (from pyro-ppl) (2.2.1+cu121)
Requirement already satisfied: tqdm>=4.36 in /usr/local/lib/python3.10/dist-packages (from pyro-ppl) (4.66.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (3.13.3)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (4.1
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 38.4 MB/s eta 0:00:00
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 823.6/823.6 kB 71.0 MB/s eta 0:00:00
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 76.9 MB/s eta 0:00:00
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.7/731.7 MB 2.1 MB/s eta 0:00:00
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 410.6/410.6 MB 2.9 MB/s eta 0:00:00
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.6/121.6 MB 7.6 MB/s eta 0:00:00
Collecting nvidia-curand-cu12==10.3.2.106 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 MB 11.1 MB/s eta 0:00:00
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 MB 8.6 MB/s eta 0:00:00
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.0/196.0 MB 5.8 MB/s eta 0:00:00
Collecting nvidia-nccl-cu12==2.19.3 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.0/166.0 MB 2.4 MB/s eta 0:00:00
Collecting nvidia-nvtx-cu12==12.1.105 (from torch>=2.0->pyro-ppl)
  Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 15.2 MB/s eta 0:00:00
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-packages (from torch>=2.0->pyro-ppl) (2.2.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch>=2.0->pyro-ppl)
  Downloading nvidia_nvjitlink_cu12-12.4.127-py3-none-manylinux2014_x86_64.whl (21.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 67.6 MB/s eta 0:00:00
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch>=2.0->pyro-ppl) (2.1.5
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch>=2.0->pyro-ppl) (1.3.0)
Installing collected packages: pyro-api, nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-runti
```

```
import os
```

```
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
from sklearn import linear_model
import seaborn as sns
import torch

import pyro
import pyro.distributions as dist
from pyro.contrib.autoguide import AutoDiagonalNormal, AutoMultivariateNormal
from pyro.infer import MCMC, NUTS, HMC, SVI, Trace_ELBO
from pyro.optim import Adam, ClippedAdam

# fix random generator seed (for reproducibility of results)
np.random.seed(42)
```

```
# load csv
df = pd.read_csv('Data_Train_AllNumbers.csv')
df.head()
```

	GGGrade	Validation	Yearly_Income	Home_Status	Lend_Amount	Deprecatory_Records
0	1	0	123200	0	10260	0
1	1	0	255200	2	27916	0
2	2	2	98842	1	14535	0
3	2	0	149600	0	6840	0
4	2	0	70400	0	10260	0

Some descriptive statistics :

```
df.describe()
```

	GGGrade	Validation	Yearly_Income	Home_Status	Lend_Amount	Deprecat
count	87499.000000	87499.000000	8.749900e+04	87499.000000	87499.000000	8
mean	2.810684	1.003280	1.262678e+05	0.591778	25920.735368	
std	1.303605	0.778242	1.011390e+05	0.659477	14433.828016	
min	1.000000	0.000000	0.000000e+00	0.000000	1710.000000	
25%	2.000000	0.000000	7.656000e+04	0.000000	15048.000000	
50%	3.000000	1.000000	1.098240e+05	0.000000	23940.000000	
75%	4.000000	2.000000	1.584000e+05	1.000000	34200.000000	
max	7.000000	2.000000	8.264031e+06	4.000000	59850.000000	

The dataset contains both numerical and categorical features, where:

1. Numerical Features: Features such as yearly income, lend amount, interest charged, inquiries, etc.
2. Categorical Features: Features like validation status (e.g., not verified, source verified), home status (e.g., own, mortgage, rent), reason for loan application (e.g., car, credit card, debt consolidation), etc.

Some data preparation before carrying out the model:

```
mat = df.values
mat

X = mat[:,0:-1]
print(X)
print(X.shape)
```

```
y = mat[:,-1].astype("int")
print(y)
print(y.shape)
```

```
train_perc = 0.66 # percentage of training data
split_point = int(train_perc*len(y))
perm = np.random.permutation(len(y)) # we also randomize the dataset
ix_train = perm[:split_point]
ix_test = perm[split_point:]
X_train = X[ix_train,:]
print(X_train)
X_test = X[ix_test,:]
y_train = y[ix_train]
print(y_train)
y_test = y[ix_test]
print("num train: %d" % len(y_train))
print("num test: %d" % len(y_test))
```

```
#using sklearn first

# logistic regression model
logreg = linear_model.LogisticRegression(solver='lbfgs', multi_class='auto', C=1)
logreg.fit(X_train, y_train)

# make predictions for test set
y_hat = logreg.predict(X_test)
print("predictions:", y_hat)
print("true values:", y_test)

# evaluate prediction accuracy
print("Accuracy:", 1.0*np.sum(y_hat == y_test) / len(y_test))
```

Using Pyro next :

```
def model(X, n_cat, obs=None):
    input_dim = X.shape[1]
    alpha = pyro.sample("alpha", dist.Normal(torch.zeros(1, n_cat),
                                             5.*torch.ones(1, n_cat)).to_event())  # Prior for the bias/intercept
    beta  = pyro.sample("beta", dist.Normal(torch.zeros(input_dim, n_cat),
                                            5.*torch.ones(input_dim, n_cat)).to_event()) # Priors for the regression coeffcients

    with pyro.plate("data"):
        y = pyro.sample("y", dist.Categorical(logits=alpha + X.matmul(beta)), obs=obs)

    return y
```

```
n_cat = 2
# Prepare data for Pyro
X_train = torch.tensor(X_train).float()
y_train = torch.tensor(y_train).float()
```

Run approximate Bayesian inference using SVI:

```
# Define guide function
guide = AutoMultivariateNormal(model)

# Reset parameter values
pyro.clear_param_store()

# Define the number of optimization steps
n_steps = 40000

# Setup the optimizer
adam_params = {"lr": 0.001}
optimizer = ClippedAdam(adam_params)

# Setup the inference algorithm
elbo = Trace_ELBO(num_particles=1)
svi = SVI(model, guide, optimizer, loss=elbo)

# Do gradient steps
for step in range(n_steps):
    elbo = svi.step(X_train, n_cat, y_train)
    if step % 1000 == 0:
        print("[%d] ELBO: %.1f" % (step, elbo))
```

```
[0] ELBO: 6046823020.7
[1000] ELBO: 1828254070.0
[2000] ELBO: 486569304.1
[3000] ELBO: 233679791.2
[4000] ELBO: 113174975.4
[5000] ELBO: 150936071.8
[6000] ELBO: 39333714.9
[7000] ELBO: 207424424.2
[8000] ELBO: 183446275.3
[9000] ELBO: 59237250.5
[10000] ELBO: 27453358.5
[11000] ELBO: 36805264.0
[12000] ELBO: 43835413.8
[13000] ELBO: 10403507.1
[14000] ELBO: 19744593.6
[15000] ELBO: 30396754.5
[16000] ELBO: 33320439.7
[17000] ELBO: 23429720.4
[18000] ELBO: 20705754.5
[19000] ELBO: 13592378.9
[20000] ELBO: 14433228.0
[21000] ELBO: 23744096.4
```

```
[22000] ELBO: 22206157.1
[23000] ELBO: 41736827.7
[24000] ELBO: 10265233.6
[25000] ELBO: 6768630.7
[26000] ELBO: 22271525.2
[27000] ELBO: 9969062.3
[28000] ELBO: 9396265.8
[29000] ELBO: 10680540.6
[30000] ELBO: 9253759.0
[31000] ELBO: 10932282.9
[32000] ELBO: 18469739.2
[33000] ELBO: 12269597.4
[34000] ELBO: 13816356.7
[35000] ELBO: 5389515.5
[36000] ELBO: 6856910.4
[37000] ELBO: 5865552.3
[38000] ELBO: 5579647.4
[39000] ELBO: 4410624.9
```

```
from pyro.infer import Predictive

predictive = Predictive(model, guide=guide, num_samples=2000,
                        return_sites=("alpha", "beta"))
samples = predictive(X_train, n_cat, y_train-1)
```

```
samples_alpha = samples["alpha"].detach().squeeze()
samples_beta = samples['beta'].detach().squeeze()
```

```
alpha_hat = samples_alpha.mean(axis=0).numpy()
beta_hat = samples_beta.mean(axis=0).numpy()
```

```
# make predictions for test set
y_hat = alpha_hat + np.dot(X_test, beta_hat)
y_hat = np.argmax(y_hat, axis=1)
print("predictions:", y_hat)
print("true values:", y_test)

# evaluate prediction accuracy
print("Accuracy:", 1.0*np.sum(y_hat == y_test) / len(y_test))
```

```
predictions: [1 1 0 ... 0 0 0]
true values: [0 0 0 ... 0 0 0]
Accuracy: 0.5683697478991596
```