In [1]:
```
import io
import json
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import sys
from sklearn.impute import KNNImputer
from sklearn import preprocessing
```

In [2]:
```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

This data, generated by the U.S. Department of Health and Human Services, dates back to 1997.

This data surveys all known public and private substance abuse treatment facilities (numbering right around 16,000) in the United States about facility characteristics as well as client demographics in order to inform the Substance Abuse and Mental Health Services Administration's assessment of available treatment services and resource requirement forecasting. This is directly applicable to our research question about the efficacy of opioid overdose prevention policies, as many of these facilities receive funding from the government as part of said policies. This survey data will allow us to see how many facilities did receive government funding, as well as how many individuals with opioid related issues were served.

Strengths of this data include:

- Abundance of features
- Abundance of datapoints
- Codebooks are satisfactory in description
- Not null-dominanted data

Weaknesses include:

- While not null-dominated, nulls still need to be dealt with.
- Some of the features are not normally-distributed.

This data was collected ethically, to the best of our knowledge. The data contains no personal identifiers that would violate an individual's privacy. Data was collected through a secure web-based questionnaire, a paper questionnaire sent via mail, and a telephone interview, accounting for a diversity of technological resources and preferences, and responses were reviewed both manually and automatically for consistency; calls were made to resolve any issues. This survey saw a response rate of 90%, with 9% of responding facilities discarded due to closure or lack of substance abuse treatment services provided. Survey response was neither enforced nor incentivised.

In [3]:
```
"""
Cell Description:
This is reading in the full samhsa dataset derived from the InitialPreproces
"""

samhsa_path = "/content/gdrive/MyDrive/L123-Project-Opiates/EDA/EDA-Data/sam

print("Begin Download...")

samhsa_df = pd.read_csv(samhsa_path)

print("SAMHSA Downloaded")
```

```
Begin Download...
<ipython-input-3-c3f31defbf4f>:10: DtypeWarning: Columns (9,121,122,123,12
4,125,126,127,128,129,130,131,132,133,134,135,136,137,138,139,140,141,142,1
43,144,145,146,164,165,166,167,168,169,171,201,204,205,208,212,213,214,215,
216,217) have mixed types. Specify dtype option on import or set low_memory
=False.
  samhsa_df = pd.read_csv(samhsa_path)
SAMHSA Downloaded
```

In [4]:
```
print("Shape: ", samhsa_df.shape)
print("Feature Names: ", samhsa_df.columns.values)
#samhsa_df.head()
```

```
Shape:  (369518, 263)
Feature Names:  ['CASEID' 'STATE' 'STFIPS' 'DETOX' 'TREATMT' 'SMISEDSUD' 'O
WNERSHP'
 'FEDOWN' 'HOSPITAL' 'LOCS' 'LOC5' 'OTP' 'ASSESSMENT' 'TESTING' 'MEDICAL'
 'TRANSITION' 'RECOVERY' 'EDUCATION' 'ANCILLARY' 'OTHER_SRVC'
 'PHARMACOTHERAPIES' 'SRVC89' 'SRVC90' 'SRVC1' 'SRVC2' 'SRVC107' 'SRVC91'
 'SRVC93' 'SRVCEDCON' 'SRVCORAL' 'SRVC10' 'SRVC11' 'SRVC73' 'SRVC74'
 'SRVC14' 'SRVC15' 'SRVC16' 'SRVCMETA' 'SRVCHAV' 'SRVCHBV' 'SRVC37'
 'SRVC27' 'SRVCODED' 'SRVCOUTCM' 'SRVC97' 'SRVC102' 'SRVC39' 'SRVC38'
 'SRVC36' 'SRVCCOACH' 'SRVC24' 'SRVC104' 'SRVC99' 'SRVC100' 'SRVC105'
 'SRVC6' 'SRVC5' 'SRVC4' 'SRVC103' 'SRVCVOCED' 'SRVC49' 'SRVC96' 'SRVC50'
 'SRVC52' 'SRVC98' 'SRVC59' 'SRVC101' 'SRVC48' 'SRVC75' 'SRVC117'
 'SRVC118' 'SRVC119' 'SRVC70' 'SRVC71' 'SRVC108' 'SRVC88' 'SRVC94'
 'SRVC106' 'SRVC95' 'SRVC85' 'SRVC87' 'SRVC86' 'SRVC129' 'SRVC130'
 'SRVCMEDHIV' 'SRVCMEDHCV' 'SRVCMEDLOFE' 'SRVCMEDCLON' 'SRVC30' 'SRVC120'
 'SRVC34' 'SRVC33' 'SRVC64' 'SRVC63' 'SRVC62' 'SRVC113' 'SRVC114'
 'SRVC115' 'SRVC61' 'SRVC31' 'SRVCPAINSA' 'SRVC32' 'SRVC121' 'SRVC122'
 'SRVC116' 'SRVC35' 'CTYPE4' 'CTYPEHI1' 'CTYPEHI2' 'CTYPE7' 'CTYPERC1'
 'CTYPERC3' 'CTYPERC4' 'CTYPE1' 'CTYPE6' 'CTYPEML' 'CTYPEOP' 'CTYPE2'
 'CTYPE3' 'SIGNLANG' 'LANG' 'LANG1' 'LANG2' 'LANG3' 'LANG4' 'LANG5'
 'LANG6' 'LANG7' 'LANG8' 'LANG9' 'LANG10' 'LANG11' 'LANG12' 'LANG13'
 'LANG14' 'LANG15' 'LANG16' 'LANG17' 'LANG18' 'LANG19' 'LANG20' 'LANG21'
 'LANG22' 'LANG23' 'LANG24' 'LANG25' 'LANG26' 'SACOUN' '_12STEP' 'BRIEF'
 'COGBT' 'DIALBT' 'CONMGMT' 'MOTIVATE' 'TRAUMAC' 'ANGERM' 'MATRIXM'
 'CRVOUCHER' 'REBTHER' 'RELPREV' 'TELEMED' 'APPRCHOTH' 'NOAPPRCH'
 'DUI_DWI' 'ONLYDUI' 'ONLYOUD' 'OPIOIDDETOX' 'OPIOIDDLOFE' 'OPIOIDMAINT'
 'OPIOIDNAL' 'OPIOIDOTH' 'OPIOIDWDRAW' 'FEESCALE' 'PAYASST' 'EARMARK'
 'REVCHK1' 'REVCHK2' 'REVCHK3' 'REVCHK5' 'REVCHK8' 'REVCHK10' 'REVCHK15'
 'REVCHK17' 'ACCRED' 'LICEN' 'LICENHOS' 'LICENMH' 'LICENPH' 'LICENSED'
 'JCAHO' 'CARF' 'NCQA' 'COA' 'HFAP' 'A_PCT' 'B_PCT' 'D_PCT' 'HIBUPNUM'
 'HIMATNUM' 'HIMETNUM' 'HIVIVNUM' 'OPBUPNUM' 'OPMATNUM' 'OPMETNUM'
 'OPVIVNUM' 'RCBUPNUM' 'RCMATNUM' 'RCMETNUM' 'RCVIVNUM' 'HIDISNUM'
 'HINALNUM' 'HIACAMNUM' 'RCDISNUM' 'RCNALNUM' 'RCACAMNUM' 'OPDISNUM'
 'OPNALNUM' 'OPACAMNUM' 'HOSPBED' 'RESBED' 'H_AGE1' 'O_AGE1' 'R_AGE1'
 'T_CLI1_D' 'T_CLI1_O' 'T_CLI1_X' 'T_CLI2_D' 'T_CLI2_O' 'T_CLI2_X'
 'T_CLI3_D' 'T_CLI3_O' 'T_CLI3_X' 'T_CLI5_D' 'T_CLI5_O' 'T_CLI5_X'
 'T_CLI6_D' 'T_CLI6_O' 'T_CLI6_X' 'T_CLI7_D' 'T_CLI7_O' 'T_CLI7_X'
 'T_CLI8_D' 'T_CLI8_O' 'T_CLI8_X' 'T_CLI9_D' 'T_CLI9_O' 'T_CLI9_X'
 'T_CLI10_D' 'T_CLI10_O' 'T_CLI10_X' 'T_CLIHI_D' 'T_CLIHI_O' 'T_CLIHI_X'
 'T_CLIML_D' 'T_CLIML_O' 'T_CLIML_X' 'T_CLIOP_D' 'T_CLIOP_O' 'T_CLIOP_X'
 'T_CLIRC_D' 'T_CLIRC_O' 'T_CLIRC_X' 'Year']
```

In [5]:

```
#Below are features identified as potentially important by Ruby and Helena
ruby_helena_feats = ['OWNERSHP', 'EARMARK', 'SMISEDSUD', 'SRVCODED','OTP', '
                     'HIBUPNUM', 'HIVIVNUM', 'OPBUPNUM', 'OPMETNUM', 'OPVIVNUM
                     'FEESCALE', 'PAYASST', 'REVCHK2','REVCHK5', 'REVCHK8', 'R
                     'LICENMH', 'LICENPH', 'LICENSED']

#Below are features that should be ignored either because of irrelevance
#or because >=66% of recorded responses were null
discard_feats = ['STFIPS', 'FEDOWN', 'LOCS', 'CTYPEHI1', 'CTYPEHI2',
                 'CTYPERC1', 'CYPTERC3', 'CYPTERC4', 'LANG1', 'LANG2',
                 'LANG3', 'LANG4', 'LANG5', 'LANG6', 'LANG7', 'LANG8',
                 'LANG9', 'LANG10', 'LANG11', 'LANG12', 'LANG13', 'LANG14',
                 'LANG15', 'LANG16','LANG17', 'LANG18','LANG19', 'LANG20',
                 'LANG21', 'LANG22','LANG23', 'LANG24','LANG25', 'LANG26',
```

```
                  'ONLYDUI', 'OPIODOTH','HIBUPNUM', 'HIMATNUM',
                  'HIMETNUM', 'HIVIVNUM','RCBUPNUM', 'RCMATNUM','RCMETNUM',
                  'RCVIVNUM','HIDISNUM', 'HINALNUM','HIACAMNUM', 'RCDISNUM',
                  'RCNALNUM', 'RCACAMNUM','HOSPBED', 'RESBED','H_AGE1', 'R_AG
                  'T_CLI1_D', 'T_CLI1_O','T_CLI1_X', 'T_CLI2_D','T_CLI2_O',
                  'T_CLI2_X','T_CLI3_D', 'T_CLI3_O','T_CLI3_X', 'T_CLI5_D',
                  'T_CLI5_O', 'T_CLI5_X','T_CLI6_D', 'T_CLI6_O','T_CLI6_X',
                  'T_CLI7_D', 'T_CLI7_O', 'T_CLI7_X','T_CLI8_D', 'T_CLI8_O',
                  'T_CLI8_X', 'T_CLI9_D', 'T_CLI9_O', 'T_CLI9_X','T_CLI10_D',
                  'T_CLI10_O','T_CLI10_X', 'T_CLIHI_D', 'T_CLIHI_O', 'T_CLIHI
                  'T_CLIML_D', 'T_CLIML_O','T_CLIML_X', 'T_CLIOP_D', 'T_CLIOP
                  'T_CLIRC_D','T_CLIRC_O', 'T_CLIRC_X']

#Below are features that need further examination because their distribution
#of responses were heavily skewed (>=95% in one response)
ill_dist_feats = ['TREATMT', 'HOSPITAL', 'SRVC89', 'SRVC37', 'SRVC100',
                  'SRVC6', 'SRVC75', 'SRVC129', 'SACOUN', 'RELPREV',
                  'NOAPPRCH', 'OPIOIDDETOX', 'OPIOIDDLOFE', 'OPIOIDMAINT', '
                  'OPIOIDWDRAW','REVCHK3', 'NCQA', 'COA', 'HFAP', 'A_PCT',
                  'B_PCT', 'D_PCT','OPBUPNUM', 'OPVIVNUM','OPDISNUM', 'OPNAL
                  'OPACAMNUM', 'O_AGE1']

#All other features should be used for EDA
```

In [6]:
```
"""
Cell Description:
main_df contains: ((ruby_helena_feats) | ~(discard_feats)) & ~(ill_dist_feat
"""
undiscarded_feats = samhsa_df.columns[~samhsa_df.columns.isin(discard_feats)
undiscarded_feats.extend(ruby_helena_feats)
plus_ruby_helena = list(set(undiscarded_feats))
main_df_features = [feat for feat in plus_ruby_helena if feat not in ill_dis
main_df = samhsa_df[main_df_features]
print("main_df shape: ", main_df.shape)
#main_df.head()
```

```
main_df shape:  (369518, 150)
```

In [7]:
```
# for feat in main_df.columns.values:
#   print(main_df[feat].value_counts())
```

In [8]:
```
"""
Cell Description:
LabelEncoding the states and saving the labelencoder as an npy object
for later use
"""
le = preprocessing.LabelEncoder()
main_df['STATE'] = le.fit_transform(main_df['STATE'])
np.save('state_classes.npy', le.classes_)
```

```
<ipython-input-8-ea97fdd71ec8>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  main_df['STATE'] = le.fit_transform(main_df['STATE'])
```

In [9]:
```
"""
Cell Description:
Replacing indicators of different non-responses as np.nan values
"""
main_cols = [feat for feat in list(main_df.columns.values) if feat != "STATE
for feat in main_cols:
  main_df[feat] = main_df[feat].replace(['M', 'N', 'V', 'D'], np.nan)
  main_df[feat] = main_df[feat].astype(float)
```

```
<ipython-input-9-4188f7ffc41a>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  main_df[feat] = main_df[feat].replace(['M', 'N', 'V', 'D'], np.nan)
<ipython-input-9-4188f7ffc41a>:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  main_df[feat] = main_df[feat].astype(float)
```

In [10]:
```
"""
Cell Description:
ill_dist_df contains features in ill_dist_feats
"""
ill_dist_df = samhsa_df[ill_dist_feats]
print("ill_dist_df shape: ", ill_dist_df.shape)
# ill_dist_df.head()
```

```
ill_dist_df shape:  (369518, 29)
```

In [11]:
```
# ill_dist_df.info()
```

In [12]:
```
# ill_dist_df.describe()
```

In [13]:
```
"""
Cell Description:
Of the ill-distributed columns/features, split them up based on if they are
binary or multinomial. This cell decides which features are which.
"""
ill_dist_nonbinary = ['A_PCT', 'B_PCT', 'D_PCT', 'OPBUPNUM', 'OPVIVNUM', 'OP
                      'OPNALNUM', 'OPACAMNUM', 'O_AGE1']
ill_dist_binary = [feat for feat in ill_dist_df.columns.values if feat not i
```

In [14]:
```
"""
Cell Description:
This cell just creates two different ill-distributed datasets based on the
grouping of the features established above.
"""
ill_dist_non_df = ill_dist_df[ill_dist_nonbinary]
ill_dist_bin_df = ill_dist_df[ill_dist_binary]
```

In [15]:
```
"""
Cell Description:
Replacing indicators of different non-responses as np.nan values
"""
for feat in ill_dist_bin_df.columns.values:
  ill_dist_bin_df[feat] = ill_dist_bin_df[feat].replace(['M'], np.nan)
  ill_dist_bin_df[feat] = ill_dist_bin_df[feat].astype(float)
# ill_dist_non_df.head()
```

```
<ipython-input-15-ff489c50a9eb>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ill_dist_bin_df[feat] = ill_dist_bin_df[feat].replace(['M'], np.nan)
<ipython-input-15-ff489c50a9eb>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ill_dist_bin_df[feat] = ill_dist_bin_df[feat].astype(float)
```

In [16]:
```
# for feat in ill_dist_bin_df.columns.values:
#   print("\nFeature: ", feat)
#   print(ill_dist_bin_df[feat].value_counts()/ill_dist_bin_df[feat].size)
```

In [17]:
```
"""
Cell Description:
Replacing indicators of different non-responses as np.nan values
"""
for feat in ill_dist_non_df.columns.values:
  ill_dist_non_df[feat] = ill_dist_non_df[feat].replace(['M', 'N'], np.nan)
  ill_dist_non_df[feat] = ill_dist_non_df[feat].astype(float)
# ill_dist_non_df.head()
```

```
<ipython-input-17-f9f7c5ed269f>:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ill_dist_non_df[feat] = ill_dist_non_df[feat].replace(['M', 'N'], np.nan)
<ipython-input-17-f9f7c5ed269f>:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-doc
s/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  ill_dist_non_df[feat] = ill_dist_non_df[feat].astype(float)
```

In [18]:
```
# for feat in ill_dist_non_df.columns.values:
#   print("\nFeature: ", feat)
#   print(ill_dist_non_df[feat].value_counts()/ill_dist_non_df[feat].size)
```

In [19]:
```
"""Ignore!"""
# for i in range(2, 10):
#   print("\n\nKNN i: ", i)
#   imputer = KNNImputer(n_neighbors=i)
#   n_ill_dist_non_df = imputer.fit_transform(ill_dist_non_df)
#   for feat in n_ill_dist_non_df.columns.values:
#       print(n_ill_dist_non_df[feat].value_counts()/n_ill_dist_non_df[feat]
#   n_ill_dist_bin_df = imputer.fit_transform(ill_dist_bin_df)
#   for feat in n_ill_dist_bin_df.columns.values:
#       print(n_ill_dist_bin_df[feat].value_counts()/n_ill_dist_bin_df[feat]
```

Out[19]:
```
'Ignore!'
```

In [20]:
```
print("main_df.shape: ", main_df.shape)
# main_df.head()
```

```
main_df.shape:  (369518, 150)
```

In [21]:
```
"""
Cell Description:
Using KNNImputer to impute np.nan values for the main_df
"""
imputer = KNNImputer(missing_values = np.nan, n_neighbors=7, keep_empty_feat
new_main_df = main_df.head(0)
new_main_df.head()
for i in range(0, main_df.shape[0], 1000):
  # print("i: ", i)
  # print("main_df.shape: ", main_df.shape)
  temp_main_df = pd.DataFrame(imputer.fit_transform(main_df.iloc[i:i+1000]))
  # print("temp_main_df.shape: ", temp_main_df.shape)
  temp_main_df.columns = new_main_df.columns
  new_main_df = pd.concat([new_main_df, temp_main_df])
print("Main Imputed")
```

```
Main Imputed
```

In [22]:
```
"""
```

```
Cell Description:
Using KNNImputer to impute np.nan values for the multinomial, ill-distribute
"""
imputer = KNNImputer(missing_values = np.nan, n_neighbors=7, keep_empty_feat
new_ill_dist_non_df = ill_dist_non_df.head(0)
for i in range(0, ill_dist_non_df.shape[0], 1000):
  # print("i: ", i)
  # print("ill_dist_non_df.shape: ", ill_dist_non_df.shape)
  temp_ill_dist_non_df = pd.DataFrame(imputer.fit_transform(ill_dist_non_df.
  temp_ill_dist_non_df.columns = new_ill_dist_non_df.columns
  new_ill_dist_non_df = pd.concat([new_ill_dist_non_df, temp_ill_dist_non_df
print("Skewed, Multinomial Imputed")
```

```
Skewed, Multinomial Imputed
```

In [23]:
```
"""
Cell Description:
Using KNNImputer to impute np.nan values for the binary, ill-distributed dat
"""
imputer = KNNImputer(missing_values = np.nan, n_neighbors=7, keep_empty_feat
new_ill_dist_bin_df = ill_dist_bin_df.head(0)
for i in range(0, ill_dist_bin_df.shape[0], 1000):
  # print("i: ", i)
  # print("ill_dist_bin_df.shape: ", ill_dist_bin_df.shape)
  temp_ill_dist_bin_df = pd.DataFrame(imputer.fit_transform(ill_dist_bin_df.
  temp_ill_dist_bin_df.columns = new_ill_dist_bin_df.columns
  new_ill_dist_bin_df = pd.concat([new_ill_dist_bin_df, temp_ill_dist_bin_df
print("Skewed, Binary Imputed")
```

```
Skewed, Binary Imputed
```

In [24]:
```
"""
Cell Description:
I only keep records that have at least 70% of its features filled in,
dropping the rest.
"""
new_main_df = new_main_df.loc[:, new_main_df.isnull().mean() <= .3]
new_ill_dist_non_df = new_ill_dist_non_df.loc[:, new_ill_dist_non_df.isnull(
new_ill_dist_bin_df = new_ill_dist_bin_df.loc[:, new_ill_dist_bin_df.isnull(
```

In [25]:
```
#Downloading the dataset to csv. This is the dataset that isn't ill-distribu
new_main_df.head(10)
new_main_df.to_csv('main_samhsa.csv', index=False)
```

In [26]:
```
#Downloading the dataset to csv. This is the multinomial ill-distributed dat
new_ill_dist_non_df.head(10)
new_ill_dist_non_df.to_csv('ill_dist_non_samhsa.csv', index=False)
```

In [27]:
```
#Downloading the dataset to csv. This is the binary ill-distributed dataset
new_ill_dist_bin_df.head(10)
new_ill_dist_bin_df.to_csv('ill_dist_bin_samhsa.csv', index=False)
```