import numpy as np
import pandas as pd
import missingno as msno
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
from fancyimpute import SimpleFill, KNN, IterativeSVD, IterativeImputer

Data Scrubber

def data_scrubber(income_df, percent_good):
'''Simple data scrubber that fills the NAN columns with the mean
of the data in the columns'''
if(income_df.isna().sum().count() >= income_df.shape[0]*percent_good):
print('Erroronious Data. Look to fix')
return(income_df.fillna(income_df.mean()))
else:
return(income_df.fillna(income_df.mean()))

def join_df_weather(dfmeter_hh, dfweather_h):
'''dfmeter_hh = half hour data from meters
dfweather_h = hourly data from London
Returns a merged list on date and time'''

```
    dfmeter_hh['date_start_time'] = pd.to_datetime(dfmeter_hh['tstp']) ## standardi
    dfweather_h['date_start_time'] = pd.to_datetime(dfweather_h['time']) ## standard

    ## Would be nice to return df with hour and half hour meter data with same weat
        ## currently only returns hour incremented data
    return pd.merge(dfmeter_hh,dfweather_h, how = 'inner', left_on='date_start_time
```

def break_by_meter(df):
'''df = weather and meter data combined
Splits data by meter'''
meter_df_list = []
for meter_name in unique_meters:
meter_df_list.append(df[df['LCLid'] == meter_name])

```
return meter_df_list
```

plot Scatter_matrix()

```
## Discuss corollations
```

def plot_scatter_matrix(df):
plot1 = pd.plotting.scatter_matrix(df)
plt.savefig('images/Scatter_matrix_of_{}.png'.format(df['LCLid'][0]))

Mean energy by by block

```
## build function
## Plot heat map of blocks
```

Split all data into train and test. Only test on test

```
## Need function to read and write new files to split data
## Place in train and test file sets for easyc iterations
    ## should probably build a database (SQL) with tables by block number
```

Add cost benifit matrix from confuion matrix see this: /Documents/galvanize/Lectures/lecture_profit-curve-imbal-classes

For each block test data

```
## Set x and y values
## X is average meter data
## Y will be the weather and time (month and day) data

## split into train and test data
## Do LinearRegression with Kfolds
    ## QQ plot

## do logisticRegression with X above or below average use
    ## ROC plot

#return linear errors
# return Logisic errors
```

if **name__ == '__main**':

dfmeter = pd.read_csv("data/smart_meters_london/daily_dataset/block_0.csv")

dfweather = pd.read_csv("data/smart_meters_london/weather_daily_darksky.csv")

```
dfmeter_hh = pd.read_csv("data/smart_meters_london/halfhourly_dataset/block_0.csv")#
dfweather_h = pd.read_csv("data/smart_meters_london/weather_hourly_darksky.csv")


df_meter_weather_hourly = join_df_weather(dfmeter_hh, dfweather_h) ## joins the two
df_meter_weather_hourly['energy'] = pd.to_numeric(df_meter_weather_hourly['energy(kW
unique_meters = df_meter_weather_hourly['LCLid'].unique() ## gets unique meters in 
meter_df_list = break_by_meter(df_meter_weather_hourly)

plot_scatter_matrix(meter_df_list[0])


dfmeter = data_scrubber(dfmeter, percent_good = 0.9)
dfweather = data_scrubber(dfweather, percent_good = 0.9)
```