scenario3

April 22, 2019

In [3]:
```
import pandas as pd
from functools import reduce
import numpy as np
import impyute as impy
from sklearn.metrics import mean_squared_error,r2_score
import datawig
```

Data Preparation

In [2]:
```
def readtodf(filename, colname):
    a=filename+'.txt'
    data = pd.read_csv(a, sep=",", header=(0))
    data.columns=['a','Date',colname]
    data=data.drop('a',axis=1)
    data['Date'] = pd.to_datetime(data['Date'])
    data = data.set_index('Date')

    return data
```

In [4]:
```
target=readtodf('prod_target','Beer')
prod_1=readtodf('prod_1','Car')
prod_2=readtodf('prod_2','Steel')
eng_1=readtodf('eng_1','Gas')
eng_2=readtodf('eng_2','Electricity')
```

In [5]:
```
temp = pd.read_csv('temp.txt', sep=",", header=(0))
temp.columns=['num','year','month','Temp']
temp=temp.drop('num',axis=1)
temp['day']=1
temp['Date']=pd.to_datetime(temp[['year', 'month', 'day']])
temp=temp.drop(['year', 'month', 'day'],axis=1)
temp = temp.set_index('Date')
```

In [6]:
```
target.shape,prod_1.shape,prod_2.shape, eng_1.shape,eng_2.shape,temp.shape
```

Out[6]:
```
((435, 1), (369, 1), (435, 1), (435, 1), (435, 1), (581, 1))
```

In [7]:
```
 # creat dataframe version of merged data
dfs = [target, prod_1, prod_2, eng_1,eng_2,temp]
```

```
        df_final = reduce(lambda left,right: pd.merge(left,right,left_index=True, right_index=
        #df_final.to_csv('data_merged.csv')

        # creat numpy version of merged data
        np_final=np.array(df_final.values,dtype=np.float)
```

```
In [20]: df_final.shape
```

```
Out[20]: (435, 6)
```

```
In [19]: df_final.head()
```

```
Out[19]:
```

Date	Beer	Car	Steel	Gas	Electricity	Temp
1956-01-01	93.2	NaN	196.9	1709	1254	25.1
1956-02-01	96.0	NaN	192.1	1646	1290	25.3
1956-03-01	95.2	NaN	201.8	1794	1379	24.9
1956-04-01	77.1	NaN	186.9	1878	1346	23.9
1956-05-01	70.9	NaN	218.0	2173	1535	19.4

0.0.1 Imputation

1.Mice The Multiple Imputation by Chained Equations (MICE) method is widely used in practice, which uses chain equations to create multiple imputations for variables of diferent types.

```
In [8]: ip_mice=impy.imputation.cs.mice(np_final)
        np.savetxt('./Imputation Results/imputation_mice.csv',ip_mice,delimiter=",")
```

2.KNN

- This method uses k-nearest neighbor to fInd similar samples and imputed unobserved data by weighted average of similar observations.
- Basic idea: Impute array with a basic mean impute and then use the resulting complete array to construct a KDTree. Use this KDTree to compute nearest neighbours. After finding k nearest neighbours, take the weighted average of them. Basically, find the nearest row in terms of distance

```
In [10]: ip_knn=impy.imputation.cs.fast_knn(np_final)
         np.savetxt('./Imputation Results/imputation_knn.csv',ip_knn,delimiter=",")
```

3.DataWig

- "Deep" Learning for Missing Value Imputationin Tables with Non-Numerical Data
- Details on the underlying model can be found in Biessmann, Salinas et al. 2018

```
In [128]: #Initialize a SimpleImputer model
          imputer = datawig.SimpleImputer(
              input_columns=['Car','Steel','Gas','Electricity','Temp'], # column(s) containing
              output_column='Beer', # the column we'd like to impute values for
```

```
    output_path = 'imputer_model' # stores model data and metrics
    )

#Using LSTMs instead of bag-of-words
# data_encoder_cols = [NumericalEncoder('Car'), NumericalEncoder('Steel'),NumericalE
#                      NumericalEncoder('Electricity'),NumericalEncoder('Temp')]
# label_encoder_cols = [NumericalEncoder('Beer')]
# data_featurizer_cols = [LSTMFeaturizer('Car'), LSTMFeaturizer('Steel'),LSTMFeaturi
#                      LSTMFeaturizer('Electricity'),LSTMFeaturizer('Temp')]

# imputer = Imputer(
#     data_featurizers=data_featurizer_cols,
#     label_encoders=label_encoder_cols,
#     data_encoders=data_encoder_cols,
#     output_path='imputer_model'
# )
```

In [137]:
```
#Fit an imputer model on the train data
imputer.fit(train_df=df_final[df_final['Beer'].notnull()], num_epochs=300,learning_ra
```

```
2019-04-20 23:21:52,585 [INFO]  Assuming 5 numeric input columns: Car, Steel, Gas, Electricity
2019-04-20 23:21:52,589 [INFO]  Assuming 0 string input columns:
2019-04-20 23:21:52,593 [INFO]  No output column name provided for ColumnEncoder using Beer
2019-04-20 23:21:52,596 [INFO]  Assuming numeric output column: Beer
2019-04-20 23:21:52,599 [INFO]  Using [[cpu(0)]] as the context for training
2019-04-20 23:21:52,605 [INFO]  Detected 0 rows with missing labels                    fo
2019-04-20 23:21:52,608 [INFO]  Dropping 0/364 rows
2019-04-20 23:21:52,611 [INFO]  Detected 0 rows with missing labels                    fo
2019-04-20 23:21:52,614 [INFO]  Dropping 0/40 rows
2019-04-20 23:21:52,617 [INFO]  Train: 364, Test: 40
2019-04-20 23:21:52,619 [INFO]  Building Train Iterator with 364 elements
2019-04-20 23:21:52,637 [INFO]  Concatenating numeric columns ['Car', 'Steel', 'Gas', 'Electri
2019-04-20 23:21:52,640 [INFO]  Normalizing with StandardScaler
2019-04-20 23:21:52,646 [INFO]  Data Encoding - Encoded 365 rows of column
2019-04-20 23:21:52,651 [INFO]  Concatenating numeric columns ['Beer'] into Beer
2019-04-20 23:21:52,653 [INFO]  Normalizing with StandardScaler
2019-04-20 23:21:52,657 [INFO]  Label Encoding - Encoded 365 rows of column
2019-04-20 23:21:52,659 [INFO]  Building Test Iterator with 40 elements
2019-04-20 23:21:52,670 [INFO]  Concatenating numeric columns ['Car', 'Steel', 'Gas', 'Electri
2019-04-20 23:21:52,672 [INFO]  Normalizing with StandardScaler
2019-04-20 23:21:52,676 [INFO]  Data Encoding - Encoded 40 rows of column
2019-04-20 23:21:52,681 [INFO]  Concatenating numeric columns ['Beer'] into Beer
2019-04-20 23:21:52,685 [INFO]  Normalizing with StandardScaler
2019-04-20 23:21:52,690 [INFO]  Label Encoding - Encoded 40 rows of column
2019-04-20 23:21:52,693 [INFO]
========== start: fit model
2019-04-20 23:21:52,695 [WARNING]  Already bound, ignoring bind()
C:\Users\Jackie Li\Anaconda3\lib\site-packages\mxnet\module\base_module.py:503: UserWarning: Pa
```

```
  allow_missing=allow_missing, force_init=force_init)
2019-04-20 23:21:52,698 [WARNING]  optimizer already initialized, ignoring...
2019-04-20 23:21:52,755 [INFO]  Epoch[0] Batch [0-37]        Speed: 3709.87 samples/sec
2019-04-20 23:21:52,800 [INFO]  Epoch[0] Train-cross-entropy=0.934271
2019-04-20 23:21:52,803 [INFO]  Epoch[0] Train-Beer-accuracy=0.000000
2019-04-20 23:21:52,806 [INFO]  Epoch[0] Time cost=0.103
2019-04-20 23:21:52,848 [INFO]  Saved checkpoint to "imputer_model\model-0000.params"
2019-04-20 23:21:52,856 [INFO]  Epoch[0] Validation-cross-entropy=1.502696
2019-04-20 23:21:52,858 [INFO]  Epoch[0] Validation-Beer-accuracy=0.000000
2019-04-20 23:21:52,907 [INFO]  Epoch[1] Batch [0-37]        Speed: 4122.24 samples/sec
2019-04-20 23:21:52,958 [INFO]  Epoch[1] Train-cross-entropy=0.922294
2019-04-20 23:21:52,960 [INFO]  Epoch[1] Train-Beer-accuracy=0.000000
2019-04-20 23:21:52,962 [INFO]  Epoch[1] Time cost=0.102
2019-04-20 23:21:52,984 [INFO]  Saved checkpoint to "imputer_model\model-0001.params"
2019-04-20 23:21:52,991 [INFO]  Epoch[1] Validation-cross-entropy=1.512594
2019-04-20 23:21:52,993 [INFO]  Epoch[1] Validation-Beer-accuracy=0.000000
2019-04-20 23:21:53,046 [INFO]  Epoch[2] Batch [0-37]        Speed: 3785.68 samples/sec
2019-04-20 23:21:53,094 [INFO]  Epoch[2] Train-cross-entropy=0.911299
2019-04-20 23:21:53,096 [INFO]  Epoch[2] Train-Beer-accuracy=0.000000
2019-04-20 23:21:53,098 [INFO]  Epoch[2] Time cost=0.103
2019-04-20 23:21:53,114 [INFO]  Saved checkpoint to "imputer_model\model-0002.params"
2019-04-20 23:21:53,123 [INFO]  Epoch[2] Validation-cross-entropy=1.523891
2019-04-20 23:21:53,125 [INFO]  Epoch[2] Validation-Beer-accuracy=0.000000
2019-04-20 23:21:53,178 [INFO]  Epoch[3] Batch [0-37]        Speed: 3785.64 samples/sec
2019-04-20 23:21:53,224 [INFO]  Epoch[3] Train-cross-entropy=0.900580
2019-04-20 23:21:53,226 [INFO]  Epoch[3] Train-Beer-accuracy=0.000000
2019-04-20 23:21:53,228 [INFO]  Epoch[3] Time cost=0.102
2019-04-20 23:21:53,251 [INFO]  Saved checkpoint to "imputer_model\model-0003.params"
2019-04-20 23:21:53,258 [INFO]  No improvement detected for 3 epochs compared to 1.50269575417
2019-04-20 23:21:53,261 [INFO]  Stopping training, patience reached
2019-04-20 23:21:53,263 [INFO]
========== done (0.5714719295501709 s) fit model
2019-04-20 23:21:53,276 [INFO]  Expected calibration error: 100.0%
2019-04-20 23:21:53,282 [INFO]  Expected calibration error after calibration: 100.0%
2019-04-20 23:21:53,301 [INFO]  save metrics in imputer_model\fit-test-metrics.json
2019-04-20 23:21:53,311 [INFO]  Keeping imputer_model\model-0000.params
2019-04-20 23:21:53,314 [INFO]  Deleting imputer_model\model-0001.params
2019-04-20 23:21:53,321 [INFO]  Deleting imputer_model\model-0002.params
2019-04-20 23:21:53,325 [INFO]  Deleting imputer_model\model-0003.params
```

Out[137]: <datawig.simple_imputer.SimpleImputer at 0x28ee200f6a0>

In [138]:
```
#Impute missing values and return original dataframe with predictions
imputed = imputer.predict(df_final)
#imputed.to_csv('./Imputation Results/imputation_Datawig.csv')
```

```
2019-04-20 23:21:55,699 [INFO]  Concatenating numeric columns ['Car', 'Steel', 'Gas', 'Electri
2019-04-20 23:21:55,701 [INFO]  Normalizing with StandardScaler
```

```
2019-04-20 23:21:55,706 [INFO]  Data Encoding - Encoded 435 rows of column
2019-04-20 23:21:55,711 [INFO]  Concatenating numeric columns ['Beer'] into Beer
2019-04-20 23:21:55,714 [INFO]  Normalizing with StandardScaler
2019-04-20 23:21:55,718 [INFO]  Label Encoding - Encoded 435 rows of column
2019-04-20 23:21:55,776 [INFO]  Top-k only for CategoricalEncoder, dropping Beer, <class 'datau
2019-04-20 23:21:55,779 [INFO]  Precision filtering only for CategoricalEncoder returning
```

```
In [139]: predictions=imputed[imputed['Beer'].notnull()]
```

```
In [140]: #Calculate MSE score
          MSE = mean_squared_error(predictions['Beer'].values, predictions['Beer_imputed'].valu

          #Calculate r2 score
          r2=r2_score(predictions['Beer'].values, predictions['Beer_imputed'].values)

          MSE,r2
```

```
Out[140]: (198.71291211265773, 0.836286238218155)
```

```
In [59]: imputed_data=imputed.copy()
         imputed_data.loc['1972-09-01':'1975-02-01','Beer']=imputed.loc['1972-09-01':'1975-02-0
         imputed_data=imputed_data.drop('Beer_imputed',axis=1);
```

```
In [60]: imputed_data.head()
```

Out[60]:

Date	Beer	Car	Steel	Gas	Electricity	Temp
1956-01-01	93.2	NaN	196.9	1709	1254	25.1
1956-02-01	96.0	NaN	192.1	1646	1290	25.3
1956-03-01	95.2	NaN	201.8	1794	1379	24.9
1956-04-01	77.1	NaN	186.9	1878	1346	23.9
1956-05-01	70.9	NaN	218.0	2173	1535	19.4

```
In [54]: #Initialize a SimpleImputer model
         imputer = datawig.SimpleImputer(
             input_columns=['Beer','Steel','Gas','Electricity','Temp'], # column(s) containing
             output_column='Car', # the column we'd like to impute values for
             output_path = 'imputer_model' # stores model data and metrics
             )
```

```
In [55]: #Fit an imputer model on the train data
         imputer.fit(train_df=imputed_data[imputed_data['Car'].notnull()], num_epochs=300)
```

```
2019-04-20 22:44:42,708 [INFO]  Assuming 5 numeric input columns: Beer, Steel, Gas, Electricity
2019-04-20 22:44:42,710 [INFO]  Assuming 0 string input columns:
2019-04-20 22:44:42,712 [INFO]  No output column name provided for ColumnEncoder using Car
2019-04-20 22:44:42,713 [INFO]  Assuming numeric output column: Car
2019-04-20 22:44:42,715 [INFO]  Using [[cpu(0)]] as the context for training
```

```
2019-04-20 22:44:42,720 [INFO]  Fitting label encoder <class 'datawig.column_encoders.Numerical
2019-04-20 22:44:42,728 [INFO]  Detected 0 rows with missing labels                          fo
2019-04-20 22:44:42,730 [INFO]  Dropping 0/332 rows
2019-04-20 22:44:42,733 [INFO]  Detected 0 rows with missing labels                          fo
2019-04-20 22:44:42,735 [INFO]  Dropping 0/36 rows
2019-04-20 22:44:42,738 [INFO]  Train: 332, Test: 36
2019-04-20 22:44:42,739 [INFO]  Fitting data encoder <class 'datawig.column_encoders.Numericall
2019-04-20 22:44:42,750 [INFO]  Building Train Iterator with 332 elements
2019-04-20 22:44:42,767 [INFO]  Concatenating numeric columns ['Beer', 'Steel', 'Gas', 'Electri
2019-04-20 22:44:42,768 [INFO]  Normalizing with StandardScaler
2019-04-20 22:44:42,773 [INFO]  Data Encoding - Encoded 336 rows of column
2019-04-20 22:44:42,778 [INFO]  Concatenating numeric columns ['Car'] into Car
2019-04-20 22:44:42,779 [INFO]  Normalizing with StandardScaler
2019-04-20 22:44:42,782 [INFO]  Label Encoding - Encoded 336 rows of column
2019-04-20 22:44:42,783 [INFO]  Building Test Iterator with 36 elements
2019-04-20 22:44:42,816 [INFO]  Concatenating numeric columns ['Beer', 'Steel', 'Gas', 'Electri
2019-04-20 22:44:42,817 [INFO]  Normalizing with StandardScaler
2019-04-20 22:44:42,820 [INFO]  Data Encoding - Encoded 48 rows of column
2019-04-20 22:44:42,823 [INFO]  Concatenating numeric columns ['Car'] into Car
2019-04-20 22:44:42,825 [INFO]  Normalizing with StandardScaler
2019-04-20 22:44:42,829 [INFO]  Label Encoding - Encoded 48 rows of column
2019-04-20 22:44:42,831 [INFO]  Concatenating all 1 latent symbols
2019-04-20 22:44:42,832 [INFO]  Constructing numerical loss for column Car
2019-04-20 22:44:42,835 [INFO]  Building output symbols
2019-04-20 22:44:42,840 [INFO]
========== start: fit model
2019-04-20 22:44:42,842 [WARNING]  Already bound, ignoring bind()
2019-04-20 22:44:42,870 [INFO]  Epoch[0] Batch [0-11]        Speed: 8823.68 samples/sec
2019-04-20 22:44:42,886 [INFO]  Epoch[0] Train-cross-entropy=13.688891
2019-04-20 22:44:42,888 [INFO]  Epoch[0] Train-Car-accuracy=0.000000
2019-04-20 22:44:42,890 [INFO]  Epoch[0] Time cost=0.043
2019-04-20 22:44:42,909 [INFO]  Saved checkpoint to "imputer_model\model-0000.params"
2019-04-20 22:44:42,914 [INFO]  Epoch[0] Validation-cross-entropy=10.143172
2019-04-20 22:44:42,916 [INFO]  Epoch[0] Validation-Car-accuracy=0.000000
2019-04-20 22:44:42,940 [INFO]  Epoch[1] Batch [0-11]        Speed: 8403.51 samples/sec
2019-04-20 22:44:42,960 [INFO]  Epoch[1] Train-cross-entropy=10.606893
2019-04-20 22:44:42,962 [INFO]  Epoch[1] Train-Car-accuracy=0.000000
2019-04-20 22:44:42,963 [INFO]  Epoch[1] Time cost=0.046
2019-04-20 22:44:42,978 [INFO]  Saved checkpoint to "imputer_model\model-0001.params"
2019-04-20 22:44:42,984 [INFO]  Epoch[1] Validation-cross-entropy=10.206568
2019-04-20 22:44:42,985 [INFO]  Epoch[1] Validation-Car-accuracy=0.000000
2019-04-20 22:44:43,006 [INFO]  Epoch[2] Batch [0-11]        Speed: 9804.33 samples/sec
2019-04-20 22:44:43,022 [INFO]  Epoch[2] Train-cross-entropy=10.107667
2019-04-20 22:44:43,024 [INFO]  Epoch[2] Train-Car-accuracy=0.000000
2019-04-20 22:44:43,027 [INFO]  Epoch[2] Time cost=0.041
2019-04-20 22:44:43,048 [INFO]  Saved checkpoint to "imputer_model\model-0002.params"
2019-04-20 22:44:43,053 [INFO]  Epoch[2] Validation-cross-entropy=10.667156
2019-04-20 22:44:43,055 [INFO]  Epoch[2] Validation-Car-accuracy=0.000000
```

```
2019-04-20 22:44:43,077 [INFO]  Epoch[3] Batch [0-11]        Speed: 9810.06 samples/sec
2019-04-20 22:44:43,094 [INFO]  Epoch[3] Train-cross-entropy=9.943596
2019-04-20 22:44:43,095 [INFO]  Epoch[3] Train-Car-accuracy=0.000000
2019-04-20 22:44:43,096 [INFO]  Epoch[3] Time cost=0.040
2019-04-20 22:44:43,113 [INFO]  Saved checkpoint to "imputer_model\model-0003.params"
2019-04-20 22:44:43,118 [INFO]  No improvement detected for 3 epochs compared to 10.1431718667
2019-04-20 22:44:43,119 [INFO]  Stopping training, patience reached
2019-04-20 22:44:43,121 [INFO]
========== done (0.2812483310699463 s) fit model
2019-04-20 22:44:43,129 [INFO]  Expected calibration error: 100.0%
2019-04-20 22:44:43,136 [INFO]  Expected calibration error after calibration: 100.0%
2019-04-20 22:44:43,144 [INFO]  save metrics in imputer_model\fit-test-metrics.json
2019-04-20 22:44:43,155 [INFO]  Keeping imputer_model\model-0000.params
2019-04-20 22:44:43,157 [INFO]  Deleting imputer_model\model-0001.params
2019-04-20 22:44:43,161 [INFO]  Deleting imputer_model\model-0002.params
2019-04-20 22:44:43,164 [INFO]  Deleting imputer_model\model-0003.params
```

```
Out[55]: <datawig.simple_imputer.SimpleImputer at 0x28edbbe9860>
```

```
In [56]: #Impute missing values and return original dataframe with predictions
         imputed_car = imputer.predict(imputed_data)
         #imputed.to_csv('./Imputation Results/imputation_Datawig.csv')
```

```
2019-04-20 22:44:56,438 [INFO]  Concatenating numeric columns ['Beer', 'Steel', 'Gas', 'Electri
2019-04-20 22:44:56,439 [INFO]  Normalizing with StandardScaler
2019-04-20 22:44:56,443 [INFO]  Data Encoding - Encoded 448 rows of column
2019-04-20 22:44:56,448 [INFO]  Concatenating numeric columns ['Car'] into Car
2019-04-20 22:44:56,451 [INFO]  Normalizing with StandardScaler
2019-04-20 22:44:56,454 [INFO]  Label Encoding - Encoded 448 rows of column
2019-04-20 22:44:56,472 [INFO]  Top-k only for CategoricalEncoder, dropping Car, <class 'datawi
2019-04-20 22:44:56,473 [INFO]  Precision filtering only for CategoricalEncoder returning
```

```
In [142]: imputed_data_final=imputed_car.copy()
          imputed_data_final.loc['1956-01-01':'1961-06-01','Car']=\
          imputed_car.loc['1956-01-01':'1961-06-01']['Car_imputed'].values
          imputed_data_final=imputed_data_final.drop('Car_imputed',axis=1)
          imputed_data_final.to_csv('data_merged_final.csv');
```