Feature Engineering

Papangkorn Inkeaw, Ph.D.

Feature Enrichment

Lab 2

Adding a missing value indicator variable

Dataset: Credit Approval dataset (https://archive-beta.ics.uci.edu/ml/datasets/credit+approval)

**** The prepared dataset is avaliable in https://www2.cs.science.cmu.ac.th/courses/204371/ ****

- Import libraries

```
import pandas as pd
from feature_engine.imputation import MeanMedianImputer
from feature_engine.imputation import CategoricalImputer
from feature_engine.imputation import AddMissingIndicator
```

- Read the dataset

```
data = pd.read_csv('creditApprovalUCI.csv')
```

- Check datatype of each variable

```
data.dtypes
```

- Check the percentage of missing values in each variable

```
data.isnull().mean()
```

Adding a missing value indicator variable

- Add a missing indicator to the numerical and categorical variables in a loop.

```
for var in ['A1', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8']:
    data[var + '_NA'] = np.where(data[var].isnull(), 1, 0)
    data[var + '_NA'] = np.where(data[var].isnull(), 1, 0)
```

- Add missing indicators using Feature-engine instead.
- Set up a transformer that will add binary indicators to three variables.

```
imputer = AddMissingIndicator(variables=['A2', 'A11', 'A15'])
```

- Fit the imputer to data.

```
imputer.fit(data)
```

- Add the missing indicators.

```
data = imputer.transform(data)
```

Impute missing data using mean/median

- Replace the missing values with the median in 3 numerical variables:

```
for var in ['A2', 'A3', 'A8']:
    value = data[var].median()
    data[var] = data[var].fillna(value)
```

- Replace the missing values with the mean in 2 numerical variables (using feature-engine library)
- Set up a mean imputation transformer using MeanMedianImputer()

```
median_imputer = MeanMedianImputer(imputation_method='mean', variables=['A11', 'A15'])
```

- Fit the median imputer

```
median_imputer.fit(X_train)
```

- Inspect the learned mean

```
median_imputer.imputer_dict_
```

- *Replace the missing values with the mean*

```
data = median_imputer.transform(data)
```

Impute missing data using most frequent category

- Replace the missing values with the most frequent category in 3 categorical variables
- Set up a a frequent category imputer using CategoricalImputer ()

```
mode_imputer = CategoricalImputer(imputation_method='frequent', variables=['A4', 'A5', 'A6','A7'])
```

- Fit the mode imputer

```
mode_imputer.fit(X_train)
```

- Inspect the learned frequent categories

```
mode_imputer.imputer_dict_
```

- *Replace the missing values with the mean*

```
data = mode_imputer.transform(data)
```

Replace missing values with an arbitrary number

- Import libraries

```
import pandas as pd
from feature_engine.imputation import ArbitraryNumberImputer
from feature_engine.imputation import CategoricalImputer
from feature_engine.imputation import RandomSampleImputer
```

- Read the dataset

```
data = pd.read_csv('creditApprovalUCI.csv')
```

- Check datatype of each variable

```
data.dtypes
```

- Check the percentage of missing values in each variable

```
data.isnull().mean()
```

- Find the maximum value of four numerical variables

```
data[['A2','A3', 'A8', 'A11']].max()
```

Replace missing values with an arbitrary value

- Create an imputation transformer with Feature-engine's ArbitraryNumberImputer() in order to replace any missing values with 99

```
imputer = ArbitraryNumberImputer(arbitrary_number=99, variables=['A2','A3', 'A8', 'A11’])
```

- Fit the arbitrary number imputer

```
imputer.fit(X_train)
```

- Replace the missing values with 99

```
data = imputer.transform(data)
```

Replace missing values with a bespoke category

- Set up the CategoricalVariableImputer() from Feature-engine, which replaces missing values with the “Missing” string, specifying the categorical variables to impute, and then fit the transformer

```
imputer = CategoricalImputer(variables=['A4', 'A5', 'A6','A7'])
imputer.fit(X_train)
```

- Replace the missing values

```
data = imputer.transform(data)
```

Implementing random sample imputation

- Set up RandomSamplemputer() and fit it

```
imputer = RandomSampleImputer(variables=[‘A15'])
imputer.fit(data)
```

- Replace the missing values

```
data = imputer.transform(data)
```

Detect outliers

Dataset: Boston House Prices dataset (http://lib.stat.cmu.edu/datasets/boston)

- Import libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import load_boston
```

- load the Boston House Prices dataset from scikit-learn

```
boston_dataset = load_boston()
```

- Capture three of the variables, RM, LSTAT, and CRIM, in a pandas dataframe

```
boston = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)[['RM', 'LSTAT',
'CRIM']]
```

Detect outliers

- Make a boxplot of the RM variable to visualize outliers.

```
sns.distplot(boston['RM'], bins=30)
```

- Create a function to find the boundaries of a variable distribution.

```
def find_boundaries(df, variable, distance):
    IQR = df[variable].quantile(0.75) - df[variable].quantile(0.25)
    lower_boundary = df[variable].quantile(0.25) - (IQR * distance)
    upper_boundary = df[variable].quantile(0.75) + (IQR * distance)
    return upper_boundary, lower_boundary
```

- Use the function to determine the limits of the RM variable.

```
RM_upper_limit, RM_lower_limit = find_boundaries(boston, 'RM', 1.5)
print(RM_upper_limit, RM_lower_limit )
```

- Create a Boolean vector to flag the outliers in RM.

```
outliers_RM = np.where(boston['RM'] > RM_upper_limit, True, np.where(boston['RM'] < RM_lower_limit,
True, False)
```

Truncate the outliers

- Remove the outliers from the dataset.

```
boston_trimmed = boston.loc[~(outliers_RM)]
```

Perform winsorization

- Make a function to winsorize a variable to arbitrary upper and lower limits.

```
def winsorize(df, variable, upper_limit, lower_limit):
    return np.where(df[variable] > upper_limit, upper_limit,
            np.where(df[variable] < lower_limit, lower_limit, df[variable]))
```

- Winsorize the RM variable.

```
boston['RM']= winsorize(boston, 'RM', boston['RM'].quantile(0.95), boston['RM'].quantile(0.05))
```

- Import Winsorizer from Feature-engine

```
from feature_engine.outliers import Winsorizer
```

- Set up a Feature-engine Winsorizer indicating which variables we want to winsorize:

```
windsorizer = Winsorizer(capping_method='quantiles', tail='both', variables=['LSTAT'], fold=0.05)
```

- Fit windsorizer to the data.

```
windsorizer.fit(boston)
```

- Winsorize the LSTAT variables.

```
boston_t = windsorizer.transform(boston)
```

Perform winsorization

- Inspect the learned 5th percentiles.
  - *windsorizer.left_tail_caps_*
- Inspect the learned 95th percentiles.
  - *windsorizer.right_tail_caps_*

Capping the variable at arbitrary maximum and minimum values

- Set up a Winsorizer indicating which variables we want to winsorize and that we want to use the mean and standard deviation to find the limits.

```
windsorizer = Winsorizer(capping_method='gaussian', tail='both', fold=3, variables=['CRIM'])
```

- Fit windsorizer to the data.

```
windsorizer.fit(boston)
```

- Winsorize the CRIM variable.

```
boston_t = windsorizer.transform(boston_t)
```

- Inspect the learned lower and upper boundaries.

```
print(windsorizer.left_tail_caps_)
print(windsorizer.right_tail_caps_)
```

Your work!

1. Download the Thyroid Disease dataset (source: https://www.kaggle.com/datasets/yasserhessein/thyroid-disease-data-set) from https://www2.cs.science.cmu.ac.th/courses/204371/
2. Write a python program to
    1. Impute missing data in the dataset
    2. Detect and treatment outliers
3. Submit your program to assignment submission system (http://hw.cs.science.cmu.ac.th/)

Note:

- Put your name and student ID in the first cell using comment tag.
- Name your python notebook file with the pattern Lab_02_XXXXXXXXX.py (XXXXXXXXX is your student ID)

References & Study Resources

- Soledad Galli. (2020). *Python Feature Engineering Cookbook*. Packt Publishing.
- https://archive-beta.ics.uci.edu/ml/datasets/credit+approval
- http://lib.stat.cmu.edu/datasets/boston
- https://www.kaggle.com/datasets/yasserhessein/thyroid-disease-data-set