Feature Engineering

Papangkorn Inkeaw, Ph.D.

Feature Transformation

Lab 3

Feature Discretization

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
from sklearn.datasets import load_boston
from feature_engine.discretisation import EqualWidthDiscretiser
from feature_engine.discretisation import DecisionTreeDiscretiser
```

- load the Boston House Prices dataset from scikit-learn

```
boston_dataset = load_boston()
data = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
data['MEDV'] = boston_dataset.target
```

Feature Discretization

We will divide the DIS continuous variable into 10 intervals.

- Create an equal-width discretizer to a continuous variable (i.e., DIS) into 10 intervals.

```
disc = EqualWidthDiscretiser(bins=10, variables = ['DIS'])
```

- Fit the discretizer.

```
disc.fit(data)
```

- Inspect the limits of the intervals.

```
print(disc.binner_dict_)
```

- Transform the variables.

```
data = disc.transform(data)
```

Feature Discretization

We will divide the LSTAT continuous variable with arbitrary intervals.

- Create a list with the arbitrary interval limits, setting the upper limit to infinity to accommodate bigger values:

```
intervals = [0, 10, 20, 30, np.Inf]
```

- Create a list with the interval limits as labels, that is, strings:

```
labels = ['0-10', '10-20', '20-30', '>30']
```

- Discretize the LSTAT variable into the pre-defined limits.

```
data['lstat_labels'] = pd.cut(data['LSTAT'], bins=intervals, labels=labels, include_lowest=True)
```

Feature Discretization

We will divide the RM continuous variable using decision tree discretization.

- Get predictor and target variable for dataset

```
target = data['MEDV']
```

- Create a decision tree discretizer, which will optimize the maximum depth of the tree based on the negative mean square error metric using 10-fold cross-validation, for the RM variable.

```
treeDisc = DecisionTreeDiscretiser(cv=10, scoring='neg_mean_squared_error', variables=[
'RM'],regression=True, param_grid={'max_depth': [1,2,3,4]})
```

- Fit the discretizer.

```
treeDisc.fit(data, target)
```

- Transform the variables.

```
data = treeDisc.transform(data)
```

Feature Scaling

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
from sklearn.datasets import load_boston
```

- load the Boston House Prices dataset from scikit-learn

```
boston_dataset = load_boston()
data = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
data['MEDV'] = boston_dataset.target
```

Feature Scaling

- Standardize the ZN feature

```
data['ZN'] = (data['ZN']-data['ZN'].mean())/data['ZN'].std()
```

- Perform max-min normalization on the INDUS feature

```
data['INDUS'] = (data['INDUS']-data['INDUS'].min())/(data['INDUS'].max()-data['INDUS'].min())
```

Standardization:

$$x_{scaled} = \frac{x - \bar{x}}{\sigma}$$

Max-min Normalization:

$$x_{scaled} = \frac{x - \max(x)}{\max(x) - \min(x)}$$

Feature Transformation

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 scipy.stats as stats
from sklearn.datasets import load_boston
from feature_engine. transformation import PowerTransformer
```

- Load the Boston House Prices dataset from scikit-learn

```
boston_dataset = load_boston( )
data = pd.DataFrame(boston_dataset.data, columns=boston_dataset.feature_names)
data['MEDV'] = boston_dataset.target
```

Feature Transformation

- Create a function for evaluate the effect of the transformation.

```
def diagnostic_plots(df, variable):
    plt.figure(figsize=(15,6))
    plt.subplot(1, 2, 1)
    df[variable].hist(bins=30)
    plt.subplot(1, 2, 2)
    stats.probplot(df[variable], dist="norm", plot=plt)
    plt.show()
```

- Plot the distribution of the LSTAT variable.

```
diagnostic_plots(data, 'LSTAT')
```

- Preform log transformation on the LSTAT feature.

```
data['LSTAT'] = np.log(data['LSTAT'])
```

- Check the distribution of the LSTAT variable after the transformation.

```
diagnostic_plots(data, 'LSTAT')
```

Feature Transformation

- Plot the distribution of the DIS variable.

```
diagnostic_plots(data, 'DIS')
```

- Preform reciprocal transformation on the LSTAT feature.

```
data['DIS'] = np.reciprocal(data['DIS'])
```

- Check the distribution of the LSTAT variable after the transformation.

```
diagnostic_plots(data, 'DIS')
```

- Plot the distribution of the NOX variable.

```
diagnostic_plots(data, ‘NOX')
```

- Preform power transformation with exp = 0.3 on the NOX feature using feature-engine library.

```
et = PowerTransformer(variables = ['NOX'], exp=0.3)
et.fit(data)
data = et.transform(data)
```

- Check the distribution of the LSTAT variable after the transformation.

```
diagnostic_plots(data, 'DIS')
```

Categorical Variable Encoding

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.encoding import OneHotEncoder
from feature_engine.encoding import OrdinalEncoder
```

- Read the dataset

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

- Check datatype of each variable

```
data.dtypes
```

Categorical Variable Encoding

- Create an encoder for encoding A4 into k binary variables

```
ohe_enc = OneHotEncoder(variables=['A4'])
```

- Fit the encoder

```
ohe_enc.fit(data)
```

- Encode the categorical variable

```
data = ohe_enc.transform(data)
```

- Create an encoder for encoding A5 into k-1 binary variables

```
dm_enc = OneHotEncoder(variables=['A4'], drop_last=True)
```

- Fit the encoder

```
dm_enc.fit(data)
```

- Encode the categorical variable

```
data = dm_enc.transform(data)
```

Categorical Variable Encoding

Encode the A7 variable.

- Inspect all possible values of A7

```
data['A7'].unique()
```

- Create a dictionary of category to integer pairs

```
ordinal_mapping = {'v': 0, 'ff': 1, 'h': 2, 'dd': 3, 'z': 4, 'bb': 5, 'j': 6, 'Missing': 7, 'n': 8,
'o': 9}
```

- Replace the categories with numbers in the original variables

```
data['A7'] = data['A7'].map(ordinal_mapping)
```

- Make an ordinary encoder

```
le = OrdinalEncoder(encoding_method='arbitrary', variables=['A7'])
```

- Fit the encoder

```
le.fit(data)
```

- Encode the categorical variable

```
data = le.transform(data)
```

Your work!

1. Download the Adult Data Set(source: https://archive.ics.uci.edu/ml/datasets/Adult) from https://www2.cs.science.cmu.ac.th/courses/204371/
2. Investigate the dataset details from https://archive.ics.uci.edu/ml/datasets/Adult .
3. Discretize the variable age.
4. Perform a feature scaling on the variable capital.gain and capital.loss.
5. Transform the value of the hours.per.week using log transformation.
6. Encoding the variable education and sex.
7. 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_03_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://archive.ics.uci.edu/ml/datasets/Adult