Train In Data

Imputation with Feature-engine

Feature-engine

- Open-source Python library
- Feature engineering
- Feature selection
- Crazy amount of methods for feature transformation and selection.

Feature-engine and feature subsets

- Transformations can be applied to specific feature groups.

- No need to slice data or use additional classes.

fit() and transform()

- fit() ➔ learns parameters from train set
- transform() ➔ transforms data

Feature-engine works with dataframes

- Takes in and returns a Pandas dataframe
  - ➢ data analysis + deployment

Feature-engine and variables

Automatically recognises numerical, categorical and datetime variables.

Feature-engine

- https://feature-engine.readthedocs.io
- https://github.com/feature-engine/feature_engine

```
pip install feature-engine

Conda install -c conda-forge feature-engine
```

Feature-engine

Allows application of each feature engineering technique to a

subset of features

```
3
4  imputer = mdi.MeanMedianImputer(imputation_method='mean',
5                                  variables=['LotFrontage', 'MasVnrArea'])
6
7  imputer.fit(X_train)
```

```
: MeanMedianImputer(imputation_method='mean',
                    variables=['LotFrontage', 'MasVnrArea'])
```

Feature-engine

The variables to impute:

```
2
3 imputer.variables
```

```
['LotFrontage', 'MasVnrArea']
```

The learned parameters:

```
2 imputer.imputer_dict_
```

```
{'LotFrontage': 69.66866746698679, 'MasVnrArea': 103.55358898721731}
```

Mean imputation

```
imp_mean = MeanMedianImputer(
        imputation_method='mean',
        variables = [“income”, “age”],
)

imp_mean.fit(X_train)

X_train = imp_mean.transform(X_train)
X_test = imp_mean.transform(X_test)
```

Median imputation

```
imp_mean = MeanMedianImputer(
        imputation_method='median',
        variables = [“income”, “age”],
)

imp_mean.fit(X_train)

X_train = imp_mean.transform(X_train)
X_test = imp_mean.transform(X_test)
```

Arbitary number imputation

```
imp_mean = ArbitraryNumberImputer(
        arbitrary_number=99,
        variables = [“income”, “age”],
)

imp_mean.fit(X_train)

X_train = imp_mean.transform(X_train)
X_test = imp_mean.transform(X_test)
```

Arbitary number imputation

```
imp_mean = ArbitraryNumberImputer(
  imputer_dict={“income”:99, “age”:-1},
)


imp_mean.fit(X_train)


X_train = imp_mean.transform(X_train)
X_test = imp_mean.transform(X_test)
```

Frequent category imputation

```
imp_mean = CategoricalImputer(
  imputation_method="frequent",
  variables = ["color", "make"],
)


imp_mean.fit(X_train)


X_train = imp_mean.transform(X_train)
X_test = imp_mean.transform(X_test)
```

Arbitrary category imputation

```
imp_mean = CategoricalImputer(
  imputation_method="missing",
  fill_value="other",
  variables = ["color", "make"]
)

imp_mean.fit(X_train)

X_train = imp_mean.transform(X_train)
X_test = imp_mean.transform(X_test)
```

Feature-engine - advantages

- Stores the learned parameters: mean, median, mode
- Can impute feature groups without slicing the data or using an additional class
- Retains original variable names and df order.

Feature-engine - limitations

- Multiple transformers for imputation.
- Need to learn / import yet another Python library.

Train In Data

THANK YOU

www.trainindata.com