Open Source for Feature Engineering

Open-source for Feature engineering

Feature-engine

Scikit-Learn Transformers

- Missing Data Imputation
  - SimpleImputer
  - IterativeImputer

- Categorical Variable Encoding
  - OneHotEncoder
  - OrdinalEncoder

- Scalers
  - Standard Scaler
  - MinMaxScaler
  - Robust Scaler
  - A few others

- Discretisation
  - KBinsDiscretizer

- Variable Transformation
  - PowerTransformer
  - FunctionTransformer

- Variable Combination
  - Polynomial Features

- Text
  - Word Count
  - TFiDF

Feature Engine Transformers

Discretisation methods

- EqualFrequencyDiscretiser
- EqualWidthDiscretiser
- DecisionTreeDiscretiser
- ArbitraryDiscreriser

Variable Transformation methods

- LogTransformer
- ReciprocalTransformer
- PowerTransformer
- BoxCoxTransformer
- YeoJohnsonTransformer

Scikit-learn Wrapper:

- SklearnTransformerWrapper

Variable Combinations:

- MathematicalCombination
- CombineWithReferenceFeature

Imputing Methods

- MeanMedianImputer
- RandomSampleImputer
- EndTailImputer
- AddMissingIndicator
- CategoricalImputer
- ArbitraryNumberImputer
- DropMissingData

Encoding Methods

- OneHotEncoder
- OrdinalEncoder
- CountFrequencyEncoder
- MeanEncoder
- WoEEncoder
- PRatioEncoder
- RareLabelEncoder
- DecisionTreeEncoder

Outlier Handling methods

- Winsorizer
- ArbitraryOutlierCapper
- OutlierTrimmer

Category Encoders

```python
import category_encoders as ce

encoder = ce.BackwardDifferenceEncoder(cols=[...])
encoder = ce.BaseNEncoder(cols=[...])
encoder = ce.BinaryEncoder(cols=[...])
encoder = ce.CatBoostEncoder(cols=[...])
encoder = ce.HashingEncoder(cols=[...])
encoder = ce.HelmertEncoder(cols=[...])
encoder = ce.JamesSteinEncoder(cols=[...])
encoder = ce.LeaveOneOutEncoder(cols=[...])
encoder = ce.MEstimateEncoder(cols=[...])
encoder = ce.OneHotEncoder(cols=[...])
encoder = ce.OrdinalEncoder(cols=[...])
encoder = ce.SumEncoder(cols=[...])
encoder = ce.PolynomialEncoder(cols=[...])
encoder = ce.TargetEncoder(cols=[...])
encoder = ce.WOEEncoder(cols=[...])
```

contrib.scikit-learn.org/categorical-encoding/

Category Encoders
latest

Search docs

- Backward Difference Coding
- BaseN
- Binary
- CatBoost Encoder
- Hashing
- Helmert Coding
- James-Stein Encoder
- Leave One Out
- M-estimate
- One Hot
- Ordinal
- Polynomial Coding
- Sum Coding
- Target Encoder
- Weight of Evidence

Docs » Category Encoders

View page source

Category Encoders

A set of scikit-learn-style transformers for encoding categorical variables into numeric with different techniques. While ordinal, one-hot, and hashing encoders have similar equivalents in the existing scikit-learn version, the transformers in this library all share a few useful properties:

- First-class support for pandas dataframes as an input (and optionally as output)
- Can explicitly configure which columns in the data are encoded by name or index, or infer non-numeric columns regardless of input type
- Can drop any columns with very low variance based on training set optionally
- Portability: train a transformer on data, pickle it, reuse it later and get the same thing out.
- Full compatibility with sklearn pipelines, input an array-like dataset like any other transformer

Usage

install as:

```
pip install category_encoders
```

or

```
conda install -c conda-forge category_encoders
```

Pipeline with Feature-engine

```
from math import sqrt
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from sklearn.linear_model import Lasso
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.pipeline import Pipeline as pipe
from sklearn.preprocessing import MinMaxScaler

from feature_engine import categorical_encoders as ce
from feature_engine import discretisers as dsc
from feature_engine import missing_data_imputers as mdi

# load dataset
data = pd.read_csv('houseprice.csv')

# drop some variables
data.drop(labels=['YearBuilt', 'YearRemodAdd', 'GarageYrBlt', 'Id'], axis=1, inplace=True)

# categorical encoders work only with object type variables
data[discrete]= data[discrete].astype('O')

# separate into train and test sets
X_train, X_test, y_train, y_test = train_test_split(data.drop(labels=['SalePrice'], axis=1),
                                                    data.SalePrice,
                                                    test_size=0.1,
                                                    random_state=0)

# set up the pipeline
price_pipe = pipe([
    # add a binary variable to indicate missing information for the 2 variables below
    ('continuous_var_imputer', mdi.AddNaNBinaryImputer(variables = ['LotFrontage'])),

    # replace NA by the median in the 2 variables below, they are numerical
    ('continuous_var_median_imputer', mdi.MeanMedianImputer(imputation_method='median', variables = ['Lo

    # replace NA by adding the label "Missing" in categorical variables
    ('categorical_imputer', mdi.CategoricalVariableImputer(variables = categorical)),

    # disretise numerical variables using trees
    ('numerical_tree_discretiser', dsc.DecisionTreeDiscretiser(cv = 3, scoring='neg_mean_squared_error',

    # remove rare labels in categorical and discrete variables
    ('rare_label_encoder', ce.RareLabelCategoricalEncoder(tol = 0.03, n_categories=1, variables = catego

    # encode categorical and discrete variables using the target mean
    ('categorical_encoder', ce.MeanCategoricalEncoder(variables = categorical+discrete)),

    # scale features
    ('scaler', MinMaxScaler()),

    # Lasso
    ('lasso', Lasso(random_state=2909, alpha=0.005))
    ])

# train feature engineering transformers and Lasso
price_pipe.fit(X_train, np.log(y_train))

# predict
pred_train = price_pipe.predict(X_train)
pred_test = price_pipe.predict(X_test)
```

Import predefined transformers

Accommodate transformers in the pipeline

Fit the pipeline and make predictions

Pipeline

Category Encoders

```python
# Creemos pipelne
titanic_pipe = Pipeline([

    # imputación de datos ausentes - sección 4
    ('imputer_num',
     mdi.ArbitraryNumberImputer(arbitrary_number=-1,
                                variables=['age', 'fare', 'cabin_num'])),
    ('imputer_cat',
     mdi.CategoricalVariableImputer(variables=['embarked', 'cabin_cat'])),

    # codificación de variables categóricas - sección 6
    ('encoder_rare_label',
     ce.RareLabelCategoricalEncoder(tol=0.01,
                                    n_categories=6,
                                    variables=['cabin_cat'])),
    ('categorical_encoder',
     ce.OrdinalCategoricalEncoder(encoding_method='ordered',
                                  variables=['cabin_cat', 'sex', 'embarked'])),

    # máquina de potenciación de gradiente
    ('gbm', GradientBoostingClassifier(random_state=0))
])
```

Pipeline

Category Encoders

```
# train pipeline
price_pipe.fit(X_train, y_train)

# transform data
price_pipe.predict(X_train)
price_pipe.predict(X_test)


price_pipe.predict(live_data)
```

Pipeline

scikit learn

Category Encoders

```
# train pipeline
price_pipe.fit(X_train, y_train)

# transform data
price_pipe.predict(X_train)
price_pipe.predict(X_test)

price_pipe.predict(live_data)
```

Python Pickle

Thank you

www.trainindata.com