

Open Source for Feature Engineering



A Category Encoders

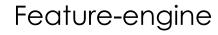


Open-source for Feature engineering















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

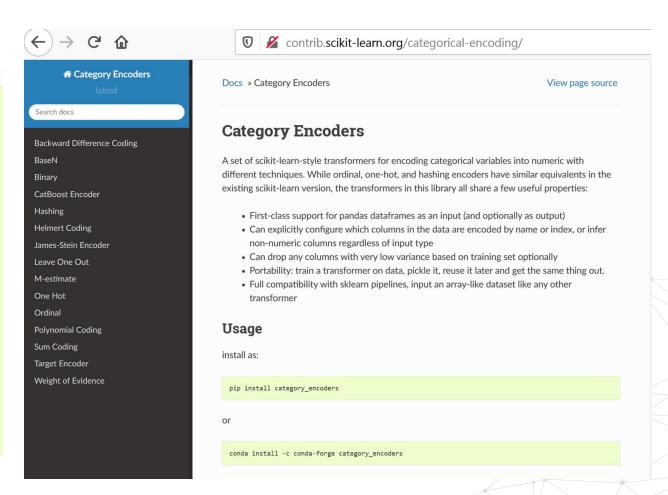
- Winsorizer
- ArbitraryOutlierCapper
- OutlierTrimmer





Category Encoders

```
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=[...])
```





Pipeline with Feature-engine

from math import sqrt

```
import pandas as pd
import numpy as np
import matplotlib.pvplot 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
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('0')
# 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 = ['Lc
   # 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 = categories=1)
   # encode categorical and discrete variables using the target mean
   ('categorical_encoder', ce.MeanCategoricalEncoder(variables = categorical+discrete)),
   # scale features
   ('scaler', MinMaxScaler()),
   ('lasso', Lasso(random state=2909, alpha=0.005))
# train feature engineering transformers and Lasso
price_pipe.fit(X_train, np.log(y_train))
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





A Category Encoders

```
# 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
    ('qbm', GradientBoostingClassifier(random state=0))
```

Pipeline



```
# 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



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)



Thank you

www.trainindata.com