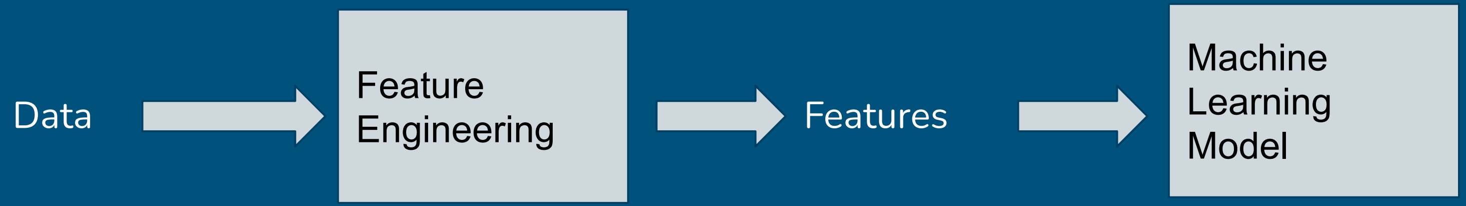
AI Saturdays Lagos Cohort 7 Practicals

Feature Engineering Automation

Feature Engineering



Feature Engineering Tools

- Feature Engine
- Featuretools

Feature engineering aspect	Scikit-learn	Feature-engine	Category encoders	Featuretools
Missing data imputation	yes	yes	no	no
Categorical encoding	yes	yes	yes	no
Discretization	yes	yes	no	no
Mathematical transformations	yes	yes	no	no
Outlier handling	no	yes	no	no
Scaling	yes	no	no	no
Text	yes	no	no	no
Transaction data	no	no	no	yes
Time Series	no	no	no	yes

Feature Engineering Tools

- Feature Engine
- Feature Tools

Transformer characteristics	Scikit-learn	Feature-engine	Category encoders
Output	NumPy array	Pandas dataframe	Pandas dataframe
Select variables	no	yes	yes
Allows Grid Search	yes	not really	no

Feature Engine

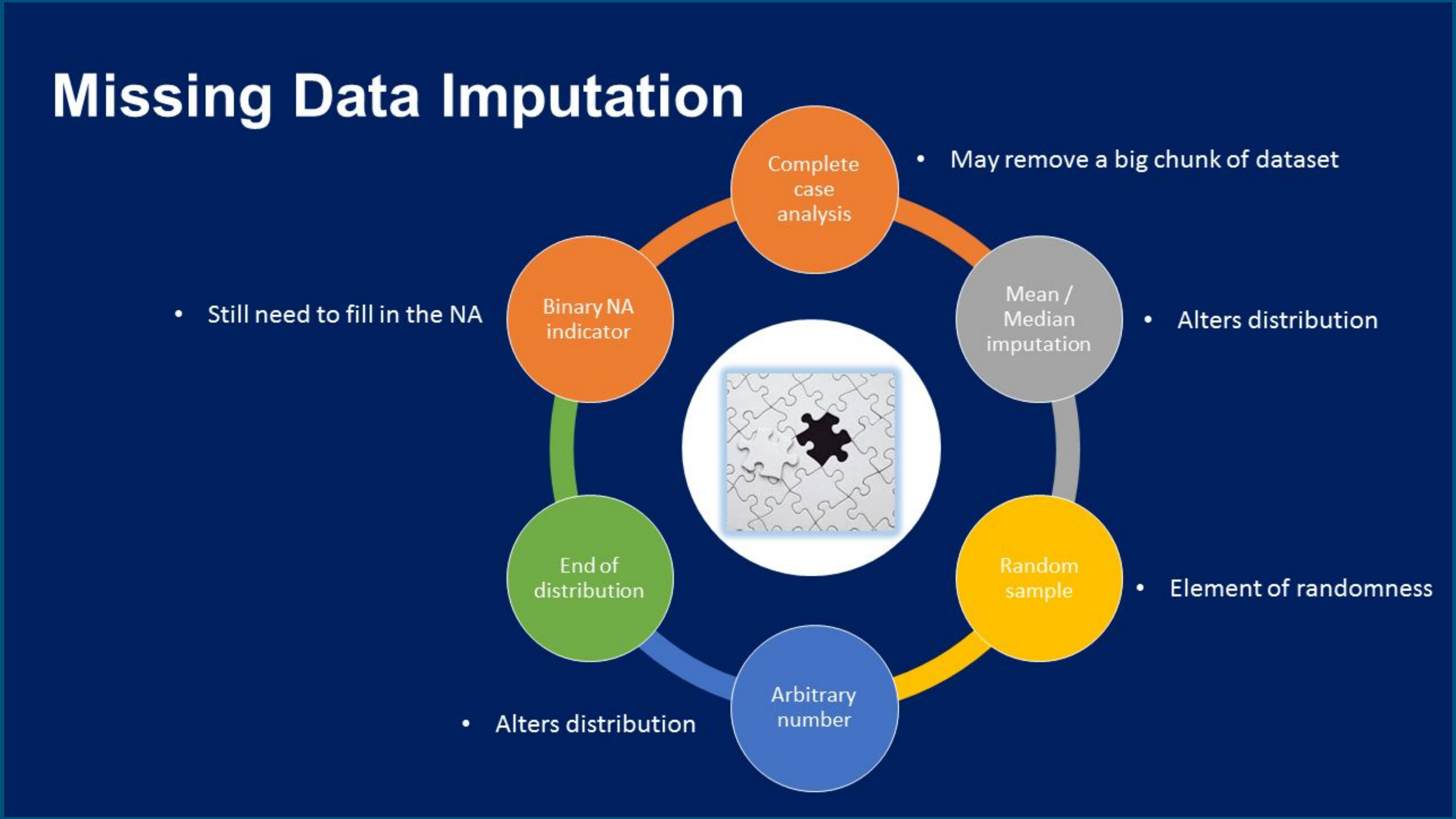
End to end feature engineering pipeline that is compatible with scikit learn.

Installation

pip install feature_engine

- Missing Data Imputation
- Categorical Encoding
- Variable Transformation
- Discretization
- Outlier Engineering
- Feature Scaling
- Date and Time Engineering
- Feature Creation

Feature Engine: Missing Data Imputation

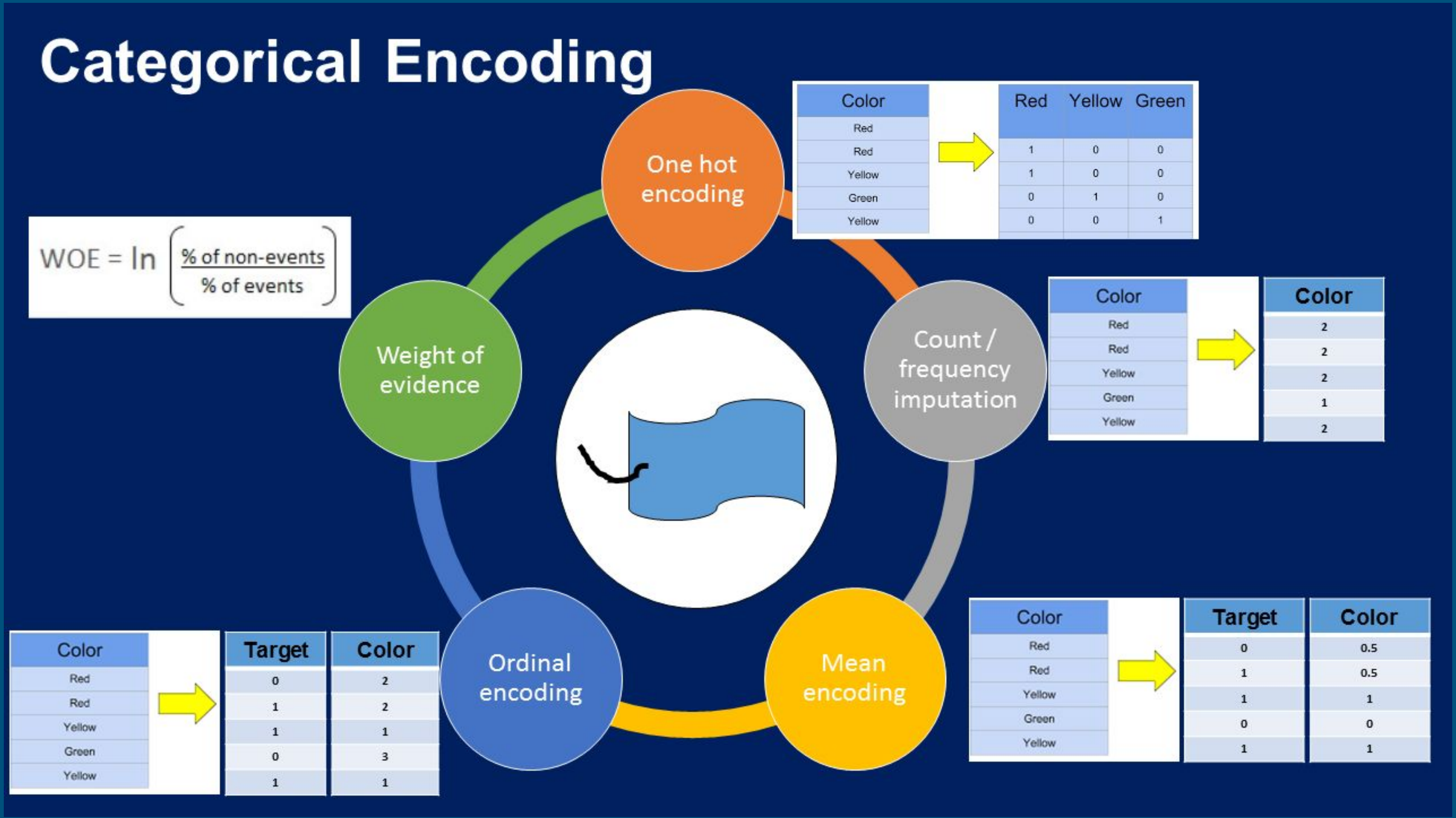


Feature Engine: Missing Data Imputation

Usage

```
from feature_engine.imputation import MeanMedianImputer

# Load dataset
data = pd.read_csv("creditApprovalUCI.csv")

# Separate into train and test sets
X_train, X_test, y_train, y_test = train_test_split(
    data.drop("A16", axis=1), data["A16"], test_size=0.3, random_state=0)

# Set up the imputer
median_imputer = MeanMedianImputer(
    imputation_method="median", variables=["A2", "A3", "A8", "A11", "A15"])
# fit the imputer
median_imputer.fit(X_train)

# transform the data
X_train = median_imputer.transform(X_train)
X_test = median_imputer.transform(X_test)
```

Feature Engine: Categorical Encoding



Color
Red
Red
Yellow
Green
Yellow

Red	Yellow	Green
1	0	0
1	0	0
0	1	0
0	0	1

Color
Red
Red
Yellow
Green
Yellow

Color
2
2
2
1
2

Color
Red
Red
Yellow
Green
Yellow

Target	Color
0	0.5
1	0.5
1	1
0	0
1	1

Color
Red
Red
Yellow
Green
Yellow

Target	Color
0	2
1	2
1	1
0	3
1	1

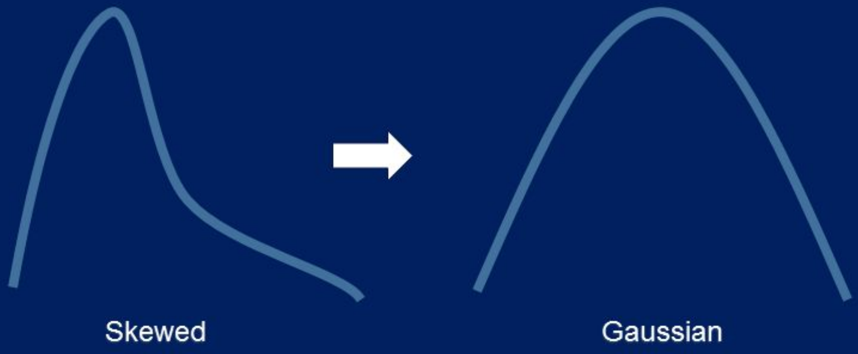
Feature Engine: Categorical Encoding

Usage

```
from feature_engine import encoding as ce

# Load dataset
    data = load_titanic()

    # Separate into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(
        data.drop(["survived", "name", "ticket"], axis=1), data["survived"], test_size=0.3,
        random_state=0,)

    # set up the encoder
    encoder = ce.CountFrequencyEncoder(
        encoding_method="frequency", variables=["cabin", "pclass", "embarked"])

    # fit the encoder
    encoder.fit(X_train)

    # transform the data
    train_t = encoder.transform(X_train)
    test_t = encoder.transform(X_test)
```

Feature Engine: Variable Transformation

Variable Transformation



Variable transformation

- Logarithmic ➔ ln(x)
- Exponential ➔ x Exp (any power)
- Reciprocal ➔ (1 / x)
- Box-Cox ➔ (x Exp (λ) – 1) / λ
  - λ varies from -5 to 5

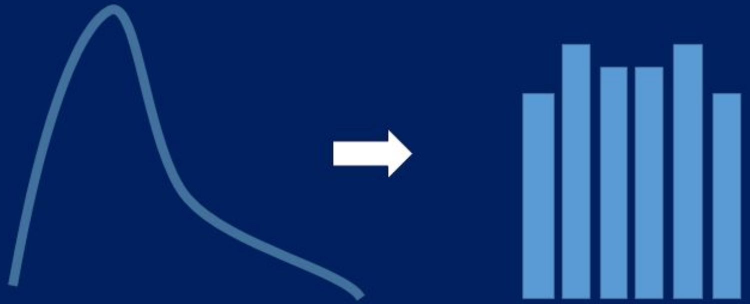
Feature Engine: Variable Transformation

Usage

```
from feature_engine import transformation as vt

# Load dataset
    data = data = pd.read_csv("houseprice.csv")

    # Separate into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(
        data.drop(["Id", "SalePrice"], axis=1),
        data["SalePrice"],
        test_size=0.3,
        random_state=0,)

    # set up the variable transformer
    tf = vt.BoxCoxTransformer(variables=["LotArea", "GrLivArea"])

    # fit the transformer
    tf.fit(X_train)

    # transform the data
    train_t = tf.transform(X_train)
    test_t = tf.transform(X_test)
```

Feature Engine: Discretization

Distribution: Discretisation



Discretisation

- Equal width bins
  - Bins ➔ (max – min) / n bins
  - Generally does not improve the spread
- Equal frequency bins
  - Bins determined by quantiles
  - Equal number of observations per bin
  - Generally improves spread

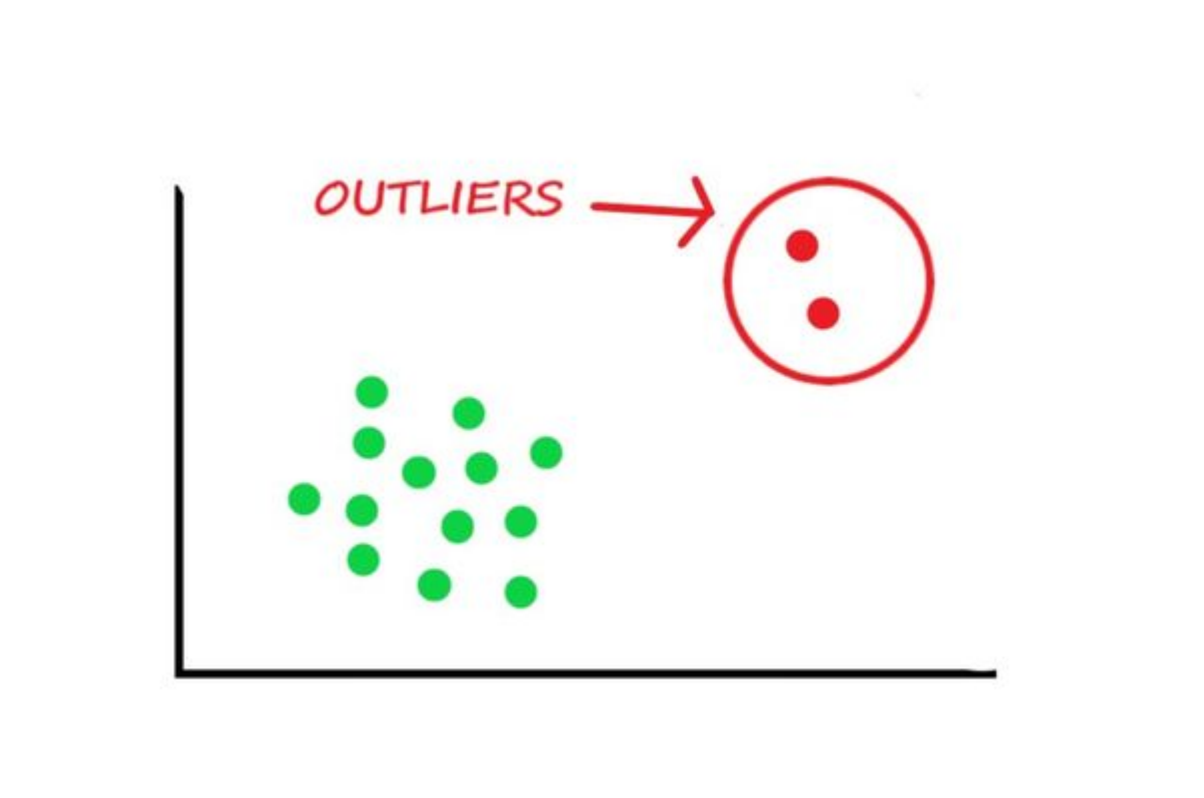
Feature Engine: Discretization

Usage

```
from feature_engine import discretisation as dsc

# Load dataset
    data = data = pd.read_csv("houseprice.csv")

    # Separate into train and test sets
    X_train, X_test, y_train, y_test = train_test_split(
        data.drop(["Id", "SalePrice"], axis=1), data["SalePrice"], test_size=0.3, random_state=0,)

    # set up the discretisation transformer
    disc = dsc.DecisionTreeDiscretiser(
        cv=3, scoring="neg_mean_squared_error", variables=["LotArea", "GrLivArea"],
        regression=True,
    )

    # fit the transformer
    disc.fit(X_train, y_train)

    # transform the data
    train_t = disc.transform(X_train)
    test_t = disc.transform(X_test)
```

Feature Engine: Outlier Engineering

- Outlier Remover
- Treating Outliers as missing values
- Winsorization (Top/ Bottom/ Zero Coding)
- Discretization



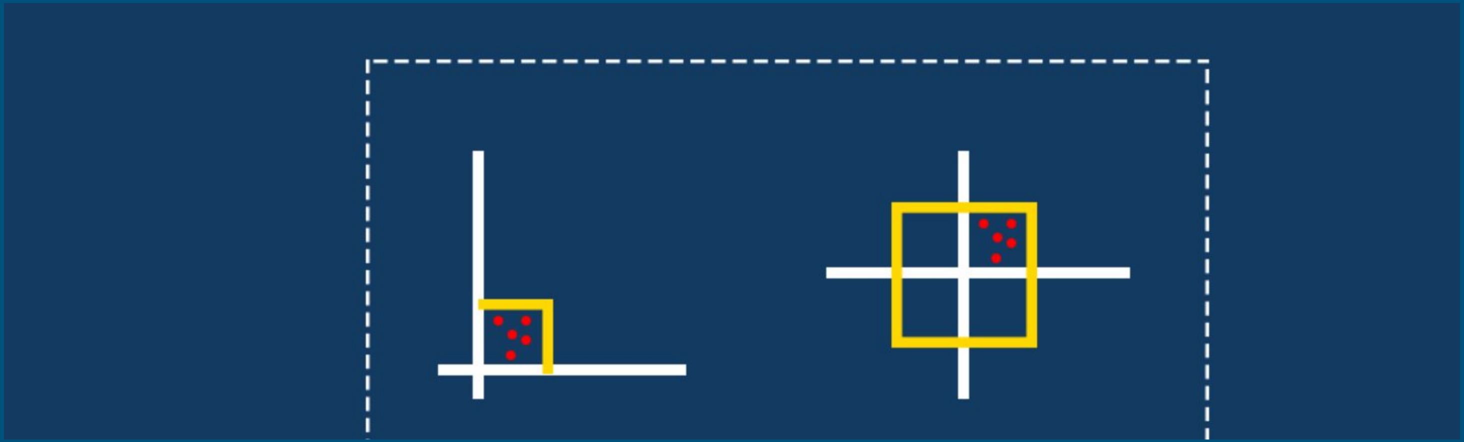
Feature Engine: Outlier Engineering

Usage

```
from feature_engine import outlier as outr

# Load dataset
   data = load_titanic2()

   # Separate into train and test sets
   X_train, X_test, y_train, y_test = train_test_split(
       data.drop(["survived", "name", "ticket"], axis=1),
       data["survived"],
       test_size=0.3, random_state=0,)

   # set up the capper
   capper = outr.Winsorizer(
       distribution="gaussian", tail="right", fold=3, variables=["age", "fare"])

   # fit the capper
   capper.fit(X_train)

   # transform the data
   train_t = capper.transform(X_train)
   test_t = capper.transform(X_test)
```

Feature Engine: Feature Scaling

- Standardization
- Min-Max Scaling
- Maximum Absolute Scaling
- Robust Scaling
- Mean normalization
- Scaling to unit length



Feature Engine: Date Time Engineering

- Year
- Month
- Day
- Day of the Week
- Is Weekend (Boolean)
- Time of the day
- Is Morning (Boolean)

Feature Engine: Feature Creation

Creating new features from existing ones.

Featuretools: Use Cases

- Predict next purchase
- Predict remaining useful life
- Predict appointment no show
- Predict loan repayment
- Predict correct answer
- Predict olympic medals
- Predict customer churn
- Predict taxi trip duration
- Predict household poverty
- Predict malicious internet traffic

References

- https://towardsdatascience.com/practical-code-implementations-of-feature-engineering-for-machine-learning-with-python-f13b953d4bcd
- https://trainindata.medium.com/feature-engine-a-new-open-source-python-package-for-feature-engineering-29a0ab88ea7c
- https://trainindata.medium.com/feature-engineering-for-machine-learning-a-comprehensive-overview-a7ad04c896f8

Feature Engineering Tools

- Auto-sklearn (drop-in replacement for sklearn)- https://automl.github.io/auto-sklearn/master/
- Sklearn-deap (evolutionary algorithms) - https://github.com/rsteca/sklearn-deap
- Tpot (automates like auto-sklearn) - https://epistasislab.github.io/tpot/
- mljar(most powerful, automation for deployment or competition with reports) - https://github.com/mljar/mljar-supervised
- feature_engine(a library for feature engineering activities) - https://github.com/solegalli/feature_engine
- Featuretools(automatically creates features from datasets) - https://www.featuretools.com