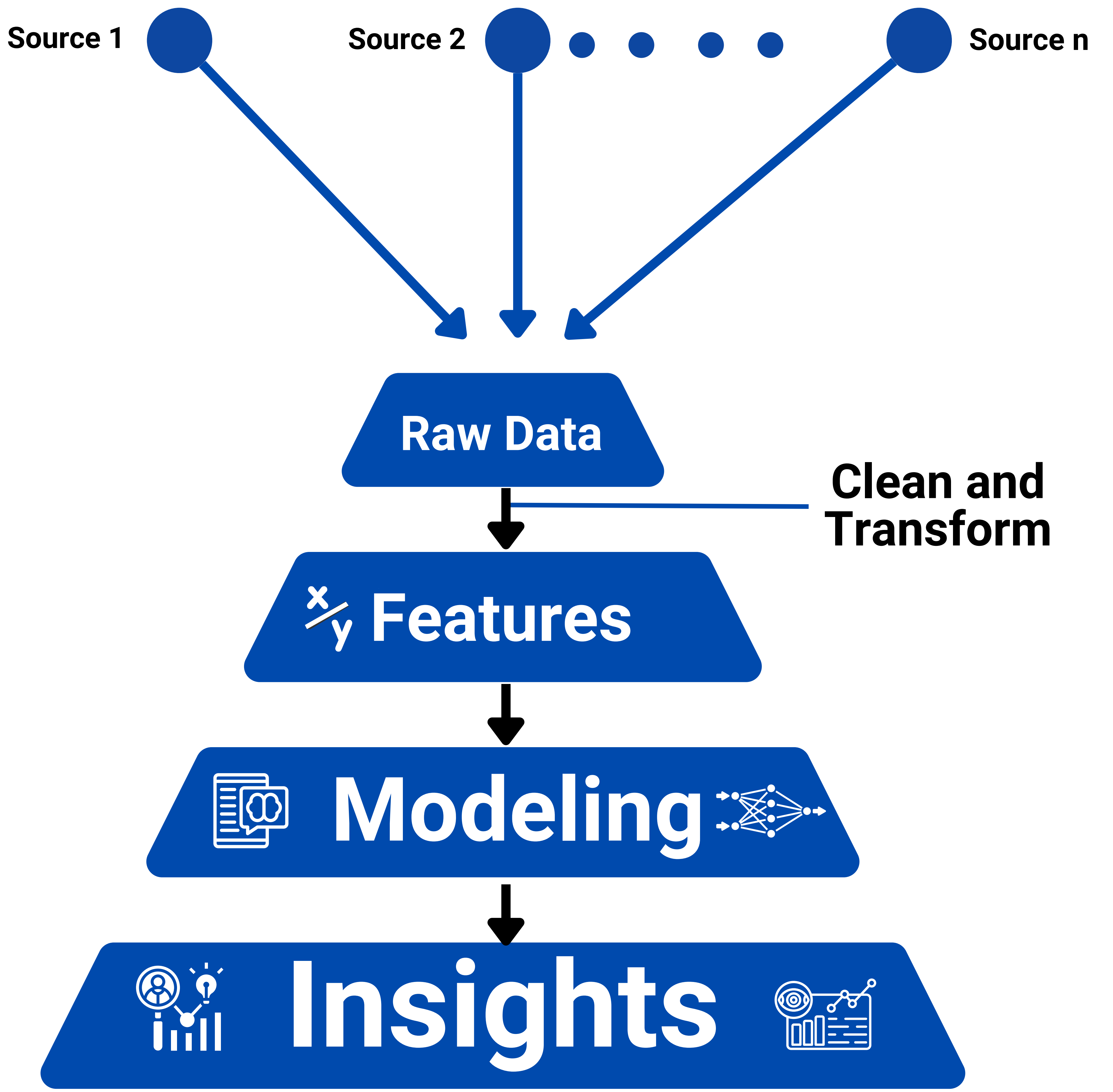
Feature Engineering in Machine Learning



Feature engineering is a critical step in the machine learning pipeline, as it can significantly affect the performance of your model. By transforming or combining features, you may be able to better capture the underlying patterns in the data.

Here are some common feature engineering techniques along with Python code snippets for each:

Missing Value Imputation

For handling missing values, you can replace them with the mean, median, or mode of the feature.

```
import pandas as pd

df = pd.DataFrame({'A': [1, 2, np.nan, 4]})
df['A'].fillna(df['A'].mean(), inplace=True)
```

One-Hot Encoding

For categorical variables, you can use one-hot encoding to convert them into numerical format.

```
df = pd.DataFrame({'B': ['red', 'green', 'blue']})
df_encoded = pd.get_dummies(df, columns=['B'])
```

Label Encoding

Another way to convert categorical variables into numerical format. Generally used when the categorical variable has some form of ordinal relationship.

```
from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()
df['B_encoded'] = le.fit_transform(df['B'])
```

Log Transformation

Useful for dealing with skewed data.

```
import numpy as np

df['A_log'] = np.log1p(df['A'])
```

Polynomial Features

This method helps to capture the interaction between features by creating polynomial terms.

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=2)
X_poly = poly.fit_transform(df[['A']])
```

Binning

Binning can be applied to continuous features to make them categorical, which might help the model capture patterns more effectively.

```
bins = [0, 1, 5, 10, np.inf]
labels = ['low', 'medium', 'high']
df['A_binned'] = pd.cut(df['A'], bins=bins, labels=labels)
```

Analytics Vidhya

Feature Scaling

Scaling features is often necessary when using algorithms that are sensitive to the scale of the input features, like SVM and k-NN.

Min-Max Scaling

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
df['A_scaled'] = scaler.fit_transform(df[['A']])
```

Standardization

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
df['A_standardized'] = scaler.fit_transform(df[['A']])
```

Analytics Vidhya

Date-Time Features

Extracting features like year, month, day, or weekday from a datetime variable can be helpful.

```
df['date'] = pd.to_datetime(df['date'])
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
```

Text Features

For text data, techniques like TF-IDF, Count Vectorization, and Word Embeddings can be used.

```
from sklearn.feature_extraction.text import TfidfVectorizer

vectorizer = TfidfVectorizer()
X_tfidf = vectorizer.fit_transform(df['text_column'])
```