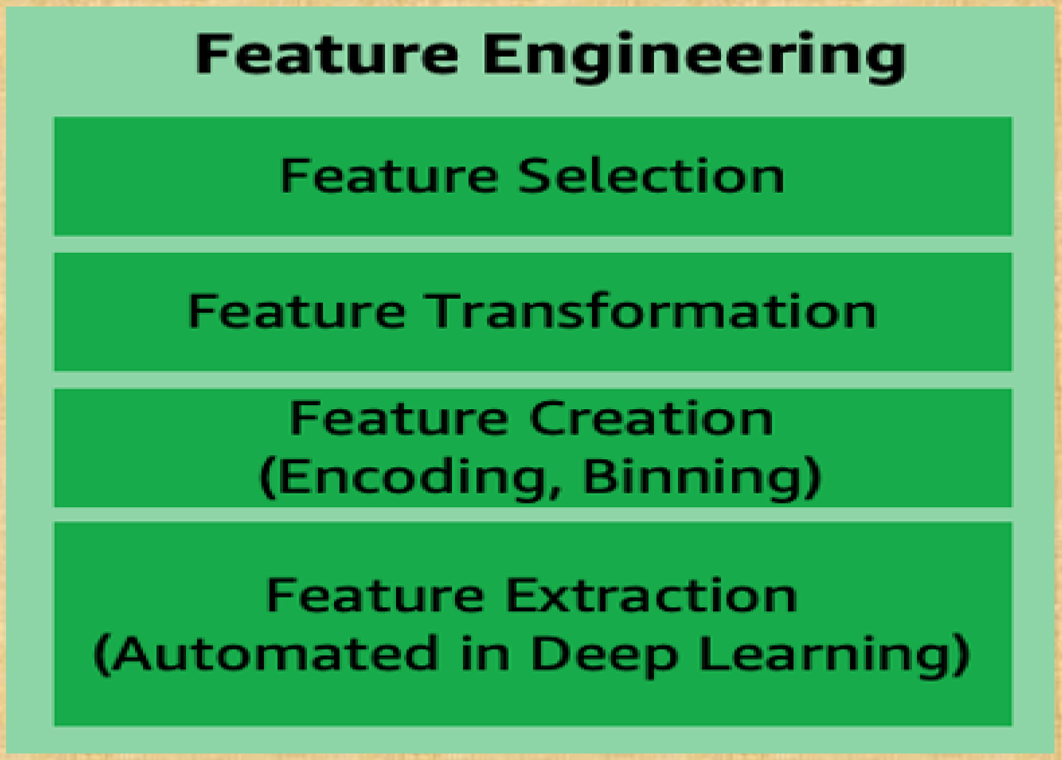

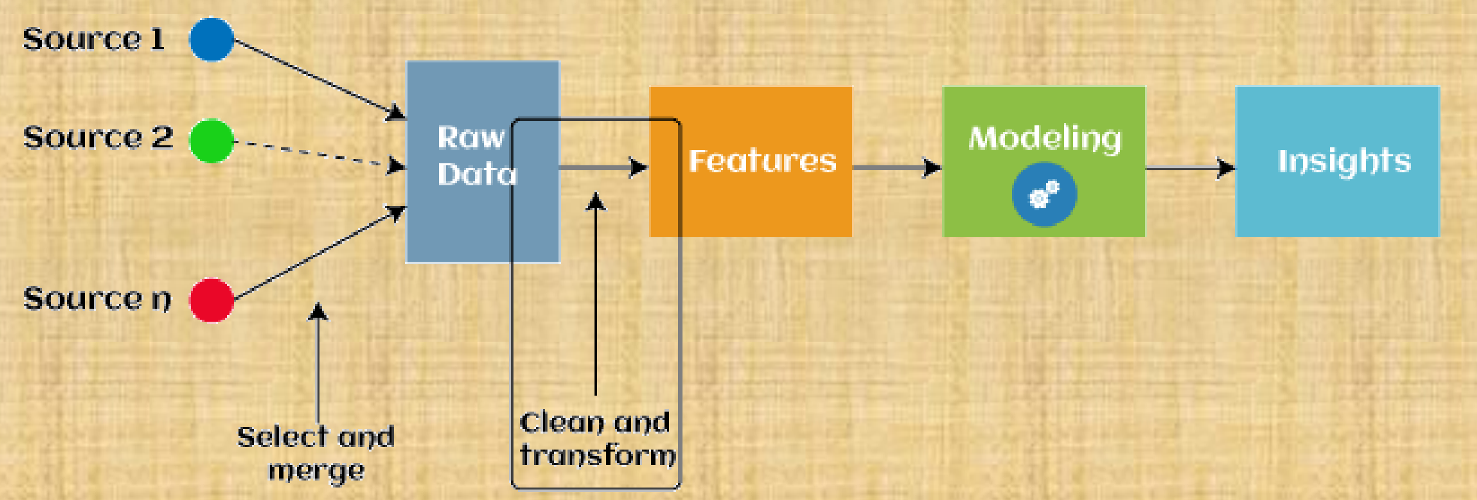

Feature Engineering

- **Definition**: The process of creating new features from existing ones to enhance model performance.
- **Experimental Nature**: It relies on the creativity of the programmer and is highly experimental.

Data Pre-processing

- **Definition**: Transforming the dataset into a form suitable for model training.
- **Common Steps**:
  - Scaling numeric variables

- Encoding categorical variables

Key Libraries for Feature Engineering and Pre-processing:

- **Pandas**
- **Scikit-learn**
- **Tsfresh**
- **Feature Engine**
- **NLTK**

Handling Extreme Values

- **Questions to Ask**:
  - Are the extreme values genuine?
  - Are the extreme values erroneous?
- **Identification Methods**:
  - Mean and Standard Deviation
  - IQR (Interquartile Range)
  - Percentiles (5th, 95th, etc.)
  - Isolation Forest, Box Plot, KDE Plot
- **Remedial Steps**:
  - Delete extreme values
  - Cap/Floor values
  - Represent as missing for later imputation

Handling Missing Values

- **Questions to Ask**:
  - Are values missing because they don't exist?
  - Were they not recorded?
- **Remedial Steps**:
  - Delete missing values
  - Impute using Mean, Median, Mode, Constant, or algorithms like MICE, KNN

Mathematical Operations

- **Combining Features**:
  - Aggregations: Sum, Max, Min, Mean
  - Relativity: Ratios, Differences
- **Transformations**:

- Log, Square Root, Reciprocal, Exponential
- Power Transformations (Box-Cox, Yeo-Johnson)

Feature Discretization:

- Equal Width, Equal Frequency, Arbitrary Intervals, K-Means Clustering

Feature Scaling

Techniques:

- Standardization
- Normalization
- Median and IQR Scaling

Handling Categorical Variables

Steps:

1. **Encoding**:
    - Ordinal Encoding
    - One-Hot Encoding
    - Rare-label Encoding
    - Frequency Encoding
    - Target Mean Encoding
2. **Manipulation**:
    - Group rare categories
    - Convert to meaningful values or binary categories

Handling Date and Time Variables

Feature Extraction:

- Day, Month, Year, Quarter, Weekday
- Hour, Minute, Second

New Features:

- Time lag, Differences between two dates

Handling Text Variables

Feature Extraction:

- Frequency of characters, words, and unique words

Ratios:

- Take ratios based on the extracted features

Feature Selection

1. **Filter Methods**:
    - Ranking variables based on statistical metrics like Chi-square, F-test, Correlation, Mutual Information.
2. **Wrapper Methods**:
    - Generate subsets of features and train models on each subset. Techniques include:
        - Exhaustive Search
        - Forward Elimination
        - Backward Elimination
3. **Embedded Methods**:
    - Use models like Linear Regression, LASSO, Decision Trees, and Random Forest to select features based on their coefficients or feature importance.

Key Components in Feature Engineering

1. **Built-in Transformers**:
    - Python Class, Python Function, Function Transformer
2. **Pipeline and Feature Union**: Combine various transformations into a cohesive process.

General Sequence of Steps for Data Preprocessing:

1. **Import Libraries**: Pandas, Scikit-learn, etc.
2. **Display Settings**: Adjust pandas display options and ignore warnings.
3. **Read Training Data**.
4. **Transformation Operations** (e.g., for specific columns like 'Airline', 'Date of Journey', etc.).
5. **Feature Selection**: Based on individual feature performance.
6. **Final Data Preprocessing and Model Training**.

Feature Engineering and Data Preprocessing

1. **Import Libraries**

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler, OneHotEncoder, PowerTransforme
from sklearn.impute import SimpleImputer
from sklearn.feature_selection import SelectKBest, chi2, mutual_info_classif
```

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import IsolationForest
import matplotlib.pyplot as plt
import seaborn as sns
```

2. **Data Overview & Display Settings**

```
pd.set_option('display.max_columns', None)
pd.set_option('display.max_rows', None)
```

3. **Reading Data**

```
# Load your dataset here
df = pd.read_csv('your_dataset.csv')
df.head()
```

Identifying and Handling Extreme Values

4. **Identify Extreme Values**
   - Using mean, standard deviation, IQR, and plots:

```
def identify_extreme_values(df, column):
  mean = df[column].mean()
  std_dev = df[column].std()
  iqr = df[column].quantile(0.75) - df[column].quantile(0.25)
  lower_bound = df[column].quantile(0.25) - 1.5 * iqr
  upper_bound = df[column].quantile(0.75) + 1.5 * iqr
  print(f'Mean: {mean}, Std Dev: {std_dev}, IQR: {iqr}')
  return lower_bound, upper_bound
```

lower, upper = identify_extreme_values(df, 'column_name')

5. **Remedial Steps for Extreme Values**
   - Capping extreme values:

```
df['column_name'] = np.where(df['column_name'] > upper, upper, df['column_n
df['column_name'] = np.where(df['column_name'] < lower, lower, df['column_n
```

6. **Visualization of Extreme Values**

```
sns.boxplot(df['column_name'])
plt.show()
```

Handling Missing Values

7. **Imputation of Missing Values**

```
imputer = SimpleImputer(strategy='mean')
df['numeric_column'] = imputer.fit_transform(df[['numeric_column']])
```

8. **Creating Indicator Columns for Missing Values**

```
df['missing_numeric_column'] = df['numeric_column'].isna().astype(int)
```

Feature Engineering

9. **Mathematical Operations on Features**

```
df['new_feature'] = df['feature1'] + df['feature2']   # Example: Sum
df['log_feature'] = np.log1p(df['numeric_column'])    # Example: Logarithmic Tran
```

10. **Polynomial Features**

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, interaction_only=True)
poly_features = poly.fit_transform(df[['feature1', 'feature2']])
```

Feature Scaling

11. **Feature Scaling**

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

Encoding Categorical Variables

12. **One-Hot Encoding**

```
ohe = OneHotEncoder()
encoded_cols = ohe.fit_transform(df[['categorical_column']]).toarray()
```

13. **Rare-Label Encoding**

```
freq_counts = df['categorical_column'].value_counts()
rare_labels = freq_counts[freq_counts < threshold].index
df['categorical_column'] = df['categorical_column'].replace(rare_labels, 'Rare')
```

Date and Time Feature Engineering

14. Extracting Date Features

```
df['day'] = pd.to_datetime(df['date_column']).dt.day
df['month'] = pd.to_datetime(df['date_column']).dt.month
```

Text Feature Engineering

15. Text Frequency Features

```
df['word_count'] = df['text_column'].apply(lambda x: len(str(x).split()))
```

Feature Selection

16. Filter Method: Chi-Square Test

```
X = df[['feature1', 'feature2']]
y = df['target']
chi2_selector = SelectKBest(chi2, k=5)
X_new = chi2_selector.fit_transform(X, y)
```

17. Wrapper Method: Recursive Feature Elimination

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression

model = LogisticRegression()
rfe = RFE(model, 5)
fit = rfe.fit(X, y)
```

Putting It All Together

18. **Data Preprocessing Pipeline**

```
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import StandardScaler, OneHotEncoder

numeric_features = ['feature1', 'feature2']
categorical_features = ['category1', 'category2']

numeric_transformer = Pipeline(steps=[
    ('scaler', StandardScaler())])

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore'))])

preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)])

clf = Pipeline(steps=[('preprocessor', preprocessor),
                      ('classifier', LogisticRegression())])

clf.fit(X_train, y_train)
```

This notebook structure will help guide the entire data preprocessing and feature engineering workflow. You can adjust specific functions and techniques based on your dataset's requirements.