1.What transformations are needed for the data?

Data transformations are crucial steps in data preprocessing to prepare raw data for analysis and modeling. These transformations help to ensure that the data meets the assumptions of statistical tests and machine learning algorithms, and to improve the performance of models. Here are some common data transformations that might be needed:

1. Handling Missing Data

Imputation: Fill missing values with mean, median, mode, or use advanced imputation techniques like KNN or regression imputation.

Deletion: Remove rows or columns with excessive missing values if imputation is not feasible.

2. Scaling and Normalization

Standardization: Transform data to have a mean of 0 and a standard deviation of 1. Suitable for algorithms that assume normally distributed data.

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

df_scaled = scaler.fit_transform(df)
```

- Min-Max Scaling: Transform data to a fixed range, typically [0, 1]. Useful for algorithms that require bounded input.

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

df_scaled = scaler.fit_transform(df)
```

- Robust Scaling: Use median and interquartile range for scaling. Useful for data with outliers.

```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

df_scaled = scaler.fit_transform(df)
```

3. Encoding Categorical Variables

- One-Hot Encoding: Convert categorical variables into binary columns.

```
df_encoded = pd.get_dummies(df, drop_first=True)
```

- Label Encoding: Assign a unique integer to each category. Useful for ordinal categorical data.

```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
```

```
df['category_column'] = encoder.fit_transform(df['category_column'])
```

4. Handling Outliers

- Clipping/Truncation: Limit the extreme values in the dataset.

```
df['column'] = df['column'].clip(lower=df['column'].quantile(0.01),
upper=df['column'].quantile(0.99))
```

- Transformation: Apply transformations like log, square root, or Box-Cox to reduce the effect of outliers.

```
import numpy as np
df['column'] = np.log1p(df['column'])  # log transformation
```

5. Feature Engineering

- Interaction Features: Create new features by combining existing ones.

```
df['new_feature'] = df['feature1'] * df['feature2']
```

- Polynomial Features: Generate polynomial and interaction features.

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree=2, include_bias=False)
df_poly = poly.fit_transform(df)
```

6. Dimensionality Reduction

- Principal Component Analysis (PCA): Reduce the dimensionality of data while preserving variance.

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
df_pca = pca.fit_transform(df)
```

- t-SNE: For visualization of high-dimensional data.

```
from sklearn.manifold import TSNE
tsne = TSNE(n_components=2)
df_tsne = tsne.fit_transform(df)
```

7. Target Variable Transformation

- Log Transformation: Transform skewed target variables to approximate a normal distribution.

```
df['target'] = np.log1p(df['target'])
```

8. Date and Time Transformations

- Extracting Date Components: Extract day, month, year, day of the week, etc.

```
df['year'] = df['date_column'].dt.year
df['month'] = df['date_column'].dt.month
df['day'] = df['date_column'].dt.day
df['day_of_week'] = df['date_column'].dt.dayofweek
```

9. Text Data Transformation

- Text Vectorization: Convert text data into numerical vectors using techniques like TF-IDF or Word2Vec.

```
from sklearn.feature_extraction.text import TfidfVectorizer
  vectorizer = TfidfVectorizer()
df_text = vectorizer.fit_transform(df['text_column'])
```

2.What are the relationships between categorical variables and numeric variables?

Categorical and numerical variables are two main types of data. Categorical variables are non-numerical information that can be divided into groups and identified by labels or names. Numerical variables are numbers that can be used to measure and analyze quantities. For example, a person's eye color, gender, or home state are categorical variables, while their height in inches is a numerical variable

Categorical variables

Use pie charts or bar graphs to show the distribution of a categorical variable. Comparative bar graphs can show associations between categorical variables.

Numerical variables

Use boxplots or histograms to show the distribution of a measurement variable. Scatterplots can illustrate associations for measurement variables.