



CoGrammar

Data Preprocessing

**SKILLS
FOR LIFE**

SKILLS BOOTCAMPS



Department
for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
(FBV: Mutual Respect.)
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes.
You can submit these questions here: [Open Class Questions](#)

Data Science Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query:
www.hyperiondev.com/support
- Report a **safeguarding** incident:
www.hyperiondev.com/safeguardreporting
- We would love your **feedback** on lectures: [Feedback on Lectures](#)

Lecture Objectives

- Understand the importance and purpose of **data preprocessing** in data science projects
- Learn and apply **advanced data preprocessing techniques** beyond basic data cleaning
- Gain hands-on experience using **Python libraries for preprocessing real-world datasets**

Lecture Objectives

- **Integrate preprocessing techniques** into machine learning workflows

Data Preprocessing

- ★ Data preprocessing is a crucial step in any data science project, **going beyond basic data cleaning** to **ensure the quality, reliability, and suitability of data** for machine learning algorithms.
- ★ Preprocessing techniques **address complex data issues**, such as **missing values, outliers, inconsistencies, and irrelevant features**, which can significantly impact the performance and generalizability of machine learning models.

Data Preprocessing

- ★ In this lesson, we will explore **advanced preprocessing techniques**, their importance, and how to apply them using Python libraries, equipping you with the skills to tackle real-world data challenges and improve the effectiveness of your machine learning projects.

Recap of Data Cleaning Concepts

- ★ Data cleaning is the process of **identifying and addressing data quality issues to ensure the accuracy, completeness, and consistency** of the dataset. It is a foundational step in data preprocessing that focuses on the following key aspects:
 - **Handling missing values:** Identifying and addressing missing data points in the dataset, either by removing instances with missing values (deletion) or imputing missing values based on statistical measures or domain knowledge (imputation).

Recap of Data Cleaning Concepts

- **Dealing with outliers:** Detecting and handling outliers, which are data points that significantly deviate from the majority of the data. Outliers can be identified using statistical methods, such as Z-score or Interquartile Range (IQR), and addressed by removal, transformation, or using robust models.

Recap of Data Cleaning Concepts

- **Resolving inconsistencies:** Identifying and resolving inconsistencies in data format, units, or conventions across different features or data sources. This ensures that the data is **standardized and comparable** for analysis and modeling.

Recap of Data Cleaning Concepts

- **Removing duplicates:** Identifying and removing duplicate records from the dataset to **avoid redundancy and potential biases in the analysis**. Duplicates can be exact matches or near-duplicates based on specific criteria.

Recap of Data Cleaning Concepts

- **Transforming data types:** Converting data types to ensure compatibility with the requirements of machine learning algorithms and to optimize computational efficiency. This includes **converting categorical variables to numerical representations and handling data type mismatches.**

Importance of Data Preprocessing

- ★ **Improved data quality:** Advanced preprocessing techniques help address data issues that **may not be captured by basic cleaning, such as handling missing data patterns, feature scaling, and encoding categorical variables.** By enhancing the quality and consistency of the data, preprocessing **lays the foundation for accurate and reliable machine learning models.**

Importance of Data Preprocessing

- ★ **Enhanced model performance:** Preprocessed data is optimized for machine learning algorithms, **enabling them to extract meaningful patterns and relationships more effectively.** Techniques like **feature scaling, encoding, and handling imbalanced data** can significantly improve model accuracy, generalization, and robustness.

Importance of Data Preprocessing

- ★ **Reduced computational complexity:** Preprocessing techniques can help **reduce the dimensionality of the data by selecting relevant features, transforming variables, and creating more efficient representations.** This leads to faster model training and inference times, making the machine learning process more computationally efficient.

Importance of Data Preprocessing

- ★ **Better interpretability and generalization:** Preprocessing steps, such as **feature engineering and selection**, can create meaningful and relevant features that align with domain knowledge. This **enhances the interpretability of the models and helps them generalize better to unseen data**, improving their practical utility in real-world applications.

Overview

- ★ **Handling missing data:** Identifying missing data patterns and mechanisms, such as **Missing Completely at Random (MCAR)**, **Missing at Random (MAR)**, and **Missing Not at Random (MNAR)**. Applying advanced imputation techniques, such as K-Nearest Neighbors (KNN) imputation, Multiple Imputation by Chained Equations (MICE), and matrix factorization, to estimate missing values based on patterns in the available data.
- ★ This process is done in either/both data cleaning and preprocessing.

Overview

- ★ **Feature scaling:** Transforming features to a **common scale to ensure fair comparison and contribution in the machine learning process**. Techniques include **standardization (Z-score normalization), min-max scaling, and robust scaling**, which handle different data distributions and outliers.

Overview

- ★ **Encoding categorical variables:** Converting **categorical variables into numerical representations** suitable for machine learning algorithms. Techniques include **one-hot encoding, label encoding, ordinal encoding, and binary encoding**, each addressing different types of categorical variables (nominal or ordinal) and handling high-cardinality scenarios.

Overview

- ★ **Feature engineering:** Creating new features from existing data to capture domain knowledge, underlying patterns, and relationships. Techniques include **polynomial features, interaction features, and domain-specific transformations.**

Overview

- ★ **Handling imbalanced data:** Addressing the challenge of imbalanced class distributions in classification tasks, **where one class (minority class) is significantly underrepresented compared to others.** Techniques include **oversampling** (random oversampling, SMOTE), **undersampling** (random undersampling, Tomek links), and **class weight adjustment**, which help balance the class distribution and improve model performance on the minority class.

Overview

- ★ **Preprocessing pipelines:** Streamlining and automating preprocessing steps into **reusable workflows** using tools like scikit-learn's Pipeline and ColumnTransformer. **Pipelines ensure consistency, reproducibility, and efficiency in applying preprocessing techniques across different datasets and machine learning tasks.**

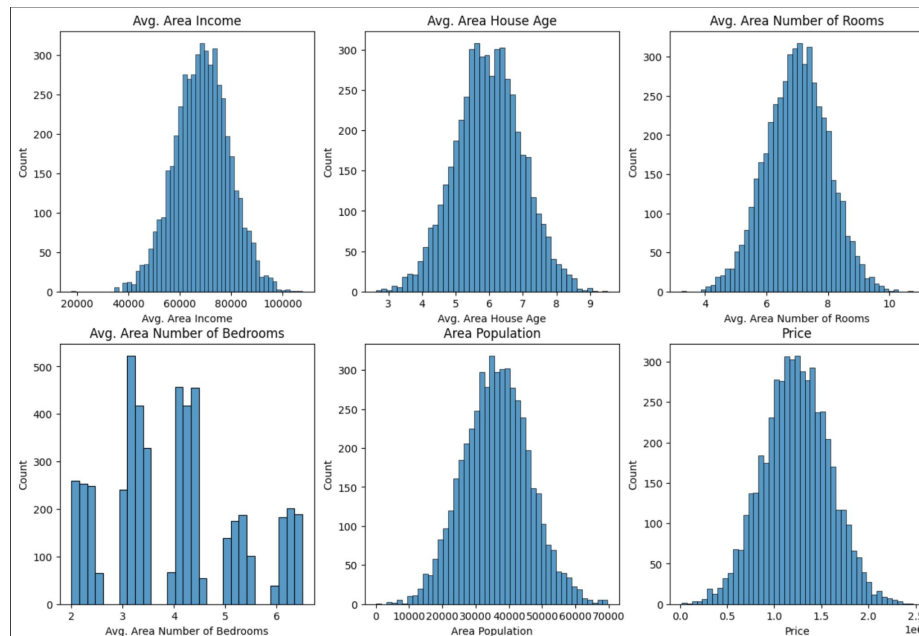
Feature Scaling

- ★ Many machine learning algorithms are **sensitive to the scale and distribution of input features**. Features with **larger values or wider ranges can dominate the learning process, leading to biased or suboptimal models**. Feature scaling ensures that all features contribute equally to the model, improving convergence, stability, and interpretability.

Feature Scaling

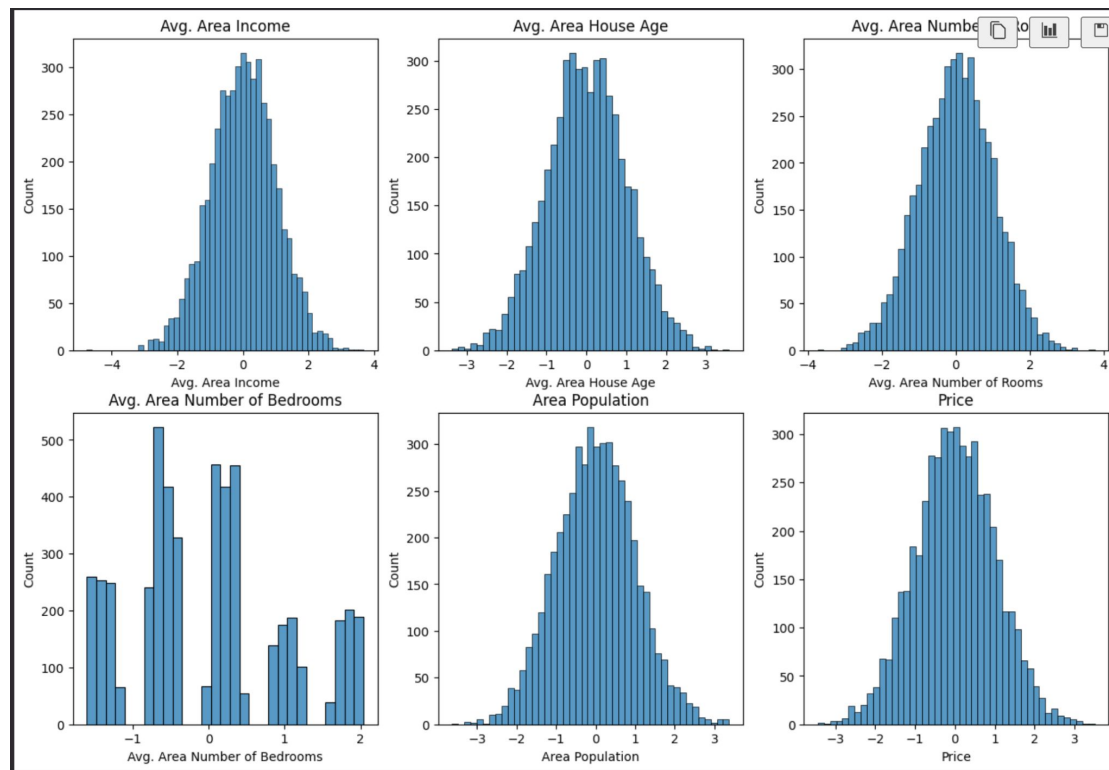
- ★ For example, in a customer churn prediction model, features like "total_spent" (ranging from **100 to 10000**) and "age" (ranging from **18 to 80**) have different scales. Scaling these features to a common range (e.g., **0 to 1**) ensures that **the model gives equal importance to both features during training.**

★ Let's take these features as examples to explain the upcoming techniques:



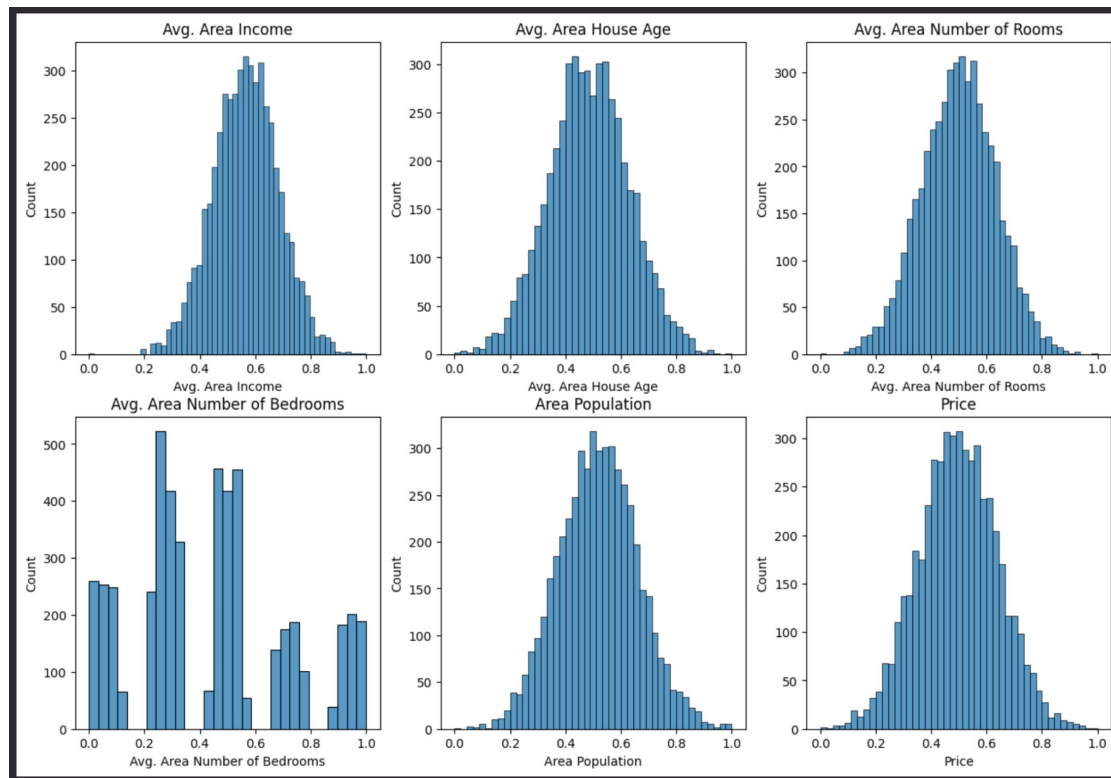
Techniques for feature scaling

- ★ **Standardization (Z-score normalization):** This technique transforms features to have **zero mean and unit variance**. It is useful when the data follows a Gaussian distribution and is sensitive to outliers. The scaled values represent the number of standard deviations away from the mean.
 - **Formula:** $(x - \text{mean}) / \text{standard deviation}$



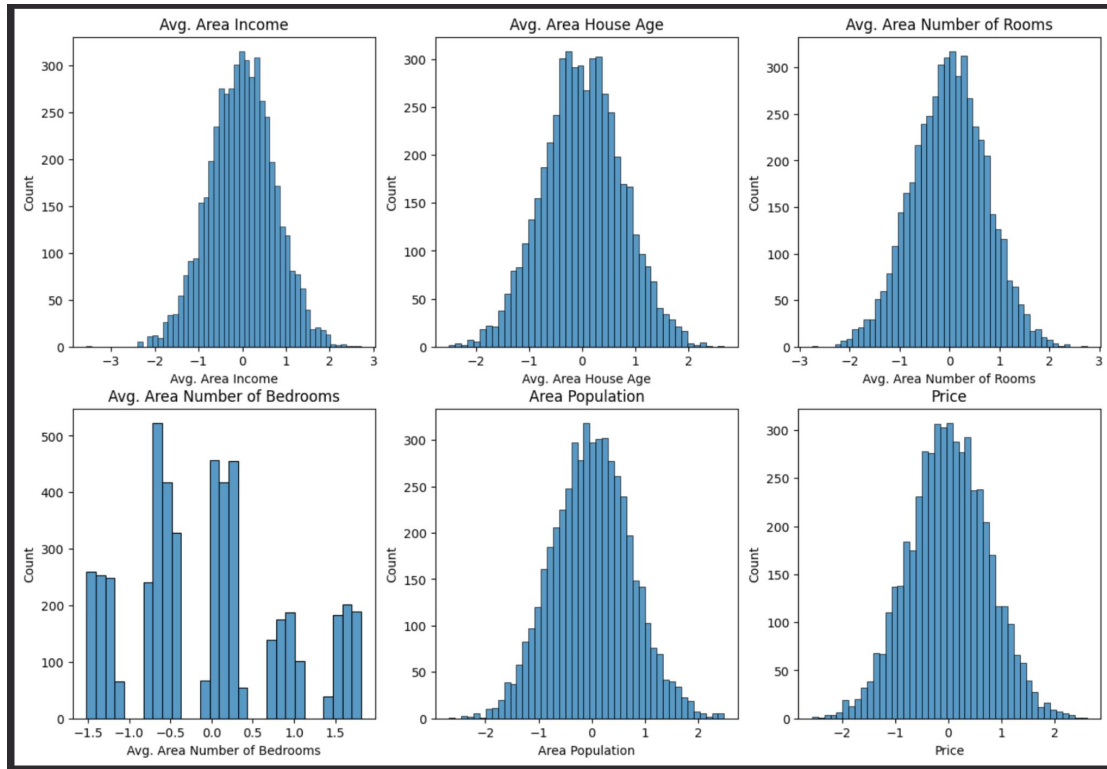
Techniques for feature scaling

- ★ **Min-Max scaling:** This technique **scales features to a specific range, typically 0 to 1**. It is useful when the data does not follow a Gaussian distribution or when bounded features are required. The scaled values represent the relative position within the original range.
 - **Formula:** $(x - \min) / (\max - \min)$



Techniques for feature scaling

- ★ **Robust scaling:** This technique uses robust statistics (median and interquartile range) to scale features, making it less sensitive to outliers. It is useful **when the data contains outliers that should not significantly impact the scaling process.**
 - Formula: $(x - \text{median}) / \text{IQR}$



Choosing the appropriate scaling technique


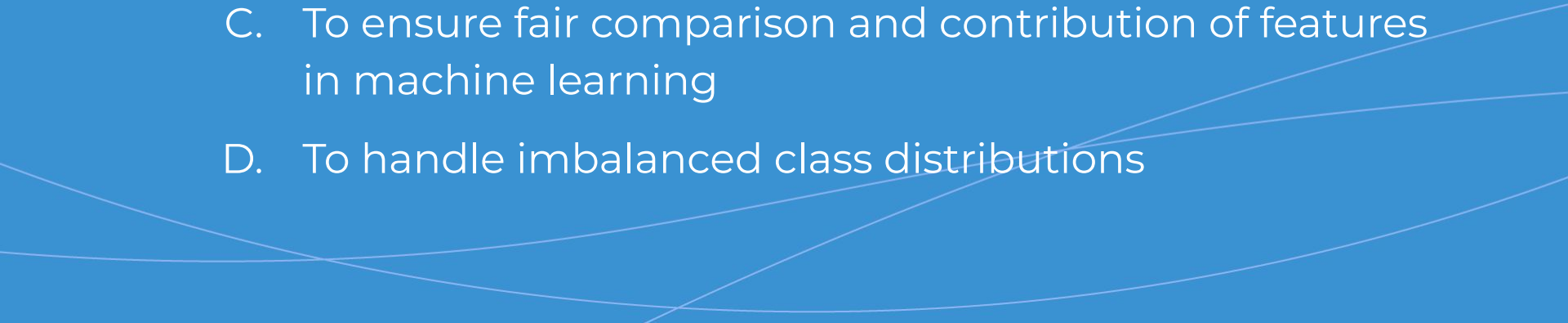
- ★ Consider the characteristics of your data, such as distribution, outliers, and domain knowledge. **Standardization is often a good default choice, while Min-Max scaling is suitable for bounded features or non-Gaussian data. Robust scaling is recommended when outliers are present but should not have a large impact on the scaled values.**

Choosing the appropriate scaling technique

- ★ Experiment with different scaling techniques and **evaluate their impact on model performance**. The choice of scaling technique can be included in a hyperparameter tuning process to find the optimal preprocessing strategy for your specific problem.





What is the purpose of feature scaling?

- 
- A. To convert categorical variables into numerical representations
 - B. To create new features from existing data
 - C. To ensure fair comparison and contribution of features in machine learning
 - D. To handle imbalanced class distributions
- 



Which feature scaling technique is less sensitive to outliers?

- 
- A. Standardization (Z-score normalization)
 - B. Min-Max scaling
 - C. Robust scaling
 - D. None of the above
- 

What is the main difference between nominal and ordinal variables?

- A. Nominal variables have categories with an inherent order, while ordinal variables do not
- B. Ordinal variables have categories with a meaningful order, while nominal variables do not
- C. Nominal and ordinal variables are the same
- D. Nominal variables are always encoded using one-hot encoding, while ordinal variables use label encoding

Let's Breathe!

**Let's take a small break
before moving on to the
next topic.**

Encoding Categorical Variables

- ★ Handling categorical variables is crucial in data preprocessing because **most machine learning algorithms require numerical inputs**. Encoding techniques **convert categorical data into numerical representations while preserving the underlying structure and relationships**. Proper encoding allows models to effectively **learn from categorical features** and capture their impact on the target variable.

Nominal vs. ordinal

- ★ Nominal variables have categories **without an inherent order or ranking**. Examples include "color" (red, blue, green) or "gender" (male, female). The numerical encoding should not imply any ordinal relationship between the categories.
- ★ Ordinal variables have categories with a **meaningful order or ranking**. Examples include "size" (small, medium, large) or "education level" (high school, bachelor's, master's, PhD). The numerical encoding should preserve the ordinal information.

Categorical

Nominal



Pen



Pencil



Paper



Dog



Cow



Cat

Ordinal



Excellent



Good



Bad



Fantastic



Okay



Don't like

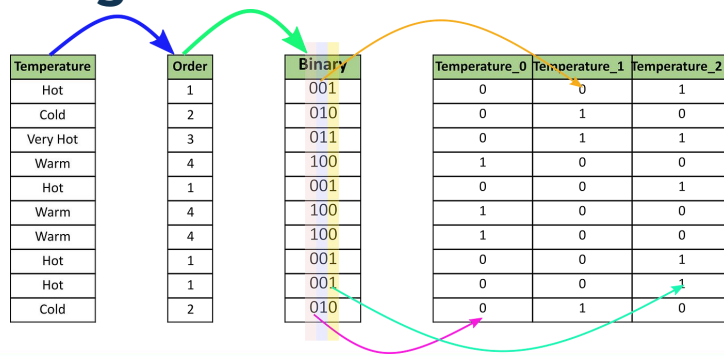
Encoding nominal variables

- ★ **One-Hot Encoding:** This technique creates **binary dummy variables for each category**. It represents the presence or absence of a category using 1s and 0s. One-hot encoding is suitable for nominal variables and is widely used due to its simplicity and interpretability. However, **it can increase the dimensionality of the feature space, especially for high-cardinality variables**.

	Sex_female	Sex_male
0	0.0	1.0
1	1.0	0.0
2	1.0	0.0
3	1.0	0.0
4	0.0	1.0

Encoding nominal variables

- ★ **Binary Encoding:** This technique **encodes categories as binary integers**, reducing the dimensionality compared to one-hot encoding. **It assigns a unique binary code to each category.** Binary encoding is useful when the number of categories is large, and the increased dimensionality from one-hot encoding is a concern.



Encoding ordinal variables

- ★ **Label Encoding:** This technique assigns numerical labels to categories **based on their order**. It is suitable for ordinal variables with a meaningful order. However, **label encoding implies a linear relationship between the encoded values, which may not always be appropriate.**



The diagram illustrates the process of label encoding for an ordinal variable. On the left, a table with the header 'Height' lists the categories 'Tall', 'Medium', and 'Short' in descending order. A light blue arrow points to the right, where the same table is shown with numerical values '0', '1', and '2' assigned to 'Tall', 'Medium', and 'Short' respectively, maintaining the original order.

Height
Tall
Medium
Short

Height
0
1
2

Encoding ordinal variables

- ★ **Ordinal Encoding:** This technique **assigns numerical labels to categories based on their order, similar to label encoding**. However, it allows for specifying custom distances between the categories to better represent the ordinal relationship.
 - **Example:** Encoding "education level" with categories "high school", "bachelor's", "master's", "PhD" as 0, 1, 3, 6 to reflect the increasing levels of education

High-cardinality variables

- ★ When dealing with categorical variables with a **large number of unique categories (high cardinality)**, traditional encoding techniques like one-hot encoding can lead to a significant increase in dimensionality. In such cases, alternative techniques can be employed:
 - **Frequency-based encoding:** Replacing categories with their frequency or occurrence count in the dataset. This technique captures the relative importance of each category.

High-cardinality variables

- ★ When dealing with categorical variables with a **large number of unique categories (high cardinality)**, traditional encoding techniques like one-hot encoding can lead to a significant increase in dimensionality. In such cases, alternative techniques can be employed:
 - **Target encoding:** Replacing categories with the mean or median of the target variable for each category. This technique incorporates the relationship between the categorical variable and the target variable.

High-cardinality variables

- ★ When dealing with categorical variables with a **large number of unique categories (high cardinality)**, traditional encoding techniques like one-hot encoding can lead to a significant increase in dimensionality. In such cases, alternative techniques can be employed:
 - **Hashing:** Applying a hash function to the categories and using the resulting hash values as numerical representations. This technique reduces dimensionality while preserving **some** information about the categories.

Feature Engineering

- ★ Feature engineering is the **process of creating new features from existing data to capture domain knowledge, underlying patterns, and relationships that can improve model performance and interpretability.** It involves transforming raw data into informative and relevant features that enhance the predictive power of machine learning models.

Importance of Feature Engineering

- ★ **Incorporating domain expertise:** Feature engineering allows data scientists to **leverage their domain knowledge and understanding of the problem** to create meaningful features that capture important aspects of the data. By incorporating domain-specific insights, models can better capture the underlying patterns and relationships.

Importance of Feature Engineering

- ★ **Improving model performance:** Well-engineered features can significantly improve the performance of machine learning models by **providing more relevant and informative inputs**. By capturing key aspects of the data and reducing noise, feature engineering can lead to better model accuracy, generalization, and robustness.

Importance of Feature Engineering

- ★ **Enhancing interpretability:** Feature engineering can create features that are **more interpretable and understandable to stakeholders**. By transforming raw data into meaningful features, models can provide insights and explanations that are easier to communicate and act upon.

Techniques Feature Engineering

- ★ **Polynomial features:** This technique generates higher-order terms from existing features to **capture non-linear relationships**. By creating polynomial combinations of features, models can learn more complex patterns and interactions.

```
housing_poly = pd.DataFrame(poly_features.fit_transform(housing_df.iloc[:, :6]),  
                             columns=poly_features.get_feature_names_out(housing_df.columns[:6]))
```

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',  
      'Avg. Area Number of Bedrooms', 'Area Population', 'Price', 'Address'],  
      dtype='object')
```

```
Index(['Avg. Area Income', 'Avg. Area House Age', 'Avg. Area Number of Rooms',  
      'Avg. Area Number of Bedrooms', 'Area Population', 'Price',  
      'Avg. Area Income^2', 'Avg. Area Income Avg. Area House Age',  
      'Avg. Area Income Avg. Area Number of Rooms',  
      'Avg. Area Income Avg. Area Number of Bedrooms',  
      'Avg. Area Income Area Population', 'Avg. Area Income Price',  
      'Avg. Area House Age^2',  
      'Avg. Area House Age Avg. Area Number of Rooms',  
      'Avg. Area House Age Avg. Area Number of Bedrooms',  
      'Avg. Area House Age Area Population', 'Avg. Area House Age Price',  
      'Avg. Area Number of Rooms^2',  
      'Avg. Area Number of Rooms Avg. Area Number of Bedrooms',  
      'Avg. Area Number of Rooms Area Population',  
      'Avg. Area Number of Rooms Price', 'Avg. Area Number of Bedrooms^2',  
      'Avg. Area Number of Bedrooms Area Population',  
      'Avg. Area Number of Bedrooms Price', 'Area Population^2',  
      'Area Population Price', 'Price^2'],  
      dtype='object')
```

Techniques Feature Engineering

- ★ **Interaction features:** This technique creates new features by **combining existing ones to capture interactions and dependencies between features**. Interaction features can reveal how the combined effect of multiple features influences the target variable.

```
housing_interact[
    'Avg. Area Bedrooms per Room'
] = housing_interact[
    'Avg. Area Number of Bedrooms'
] / housing_interact[
    'Avg. Area Number of Rooms'
]
```

Techniques Feature Engineering

- ★ **Domain-specific features:** This technique involves applying domain expertise to **create features that are specific to the problem at hand**. Domain knowledge can help identify relevant factors, derived variables, and aggregations that are meaningful for the given context.
 - Example: In a retail sales prediction problem, creating features like "days_since_last_purchase", "average_basket_value", or "product_category_preferences" based on domain understanding

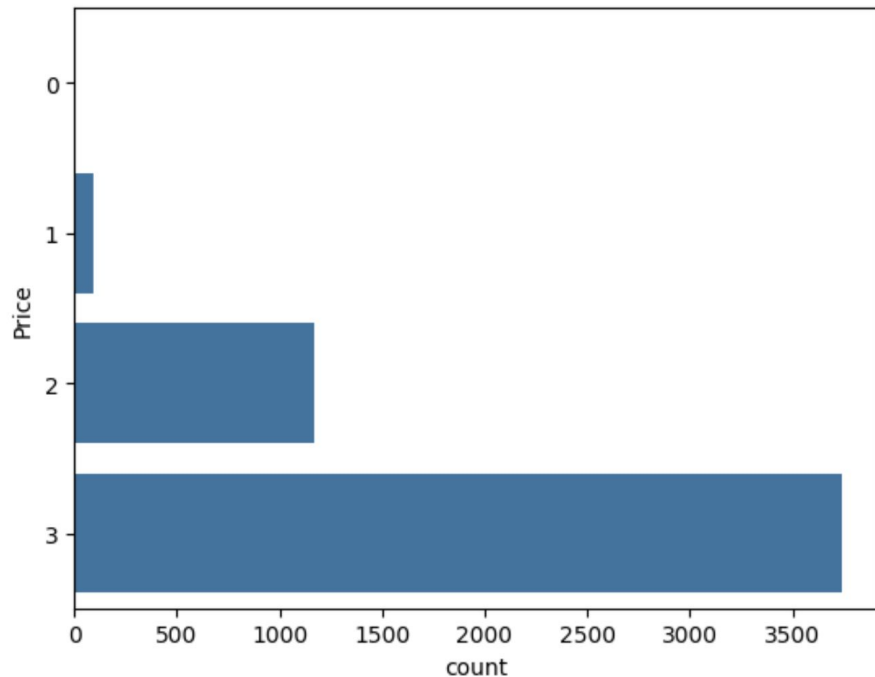
Practical considerations

- ★ It is important to **validate** the effectiveness of engineered features through **model evaluation and feature importance analysis**. By assessing the impact of each feature on model performance, data scientists can refine and iterate on their feature engineering approach to create the most informative and relevant features for the problem at hand.
- ★ We'll look at this when we get to model evaluation.

Handling Imbalanced Data

- ★ Imbalanced datasets occur when the **distribution of classes in the target variable is highly skewed, with one or more classes being significantly underrepresented compared to others.** Imbalanced data poses challenges for machine learning algorithms, as they tend to be biased towards the majority class, leading to poor performance on the minority class.

Handling Imbalanced Data



Handling Imbalanced Data

- ★ **Avoiding biased models:** When trained on imbalanced data, machine learning models can become biased towards the majority class, as they optimize for overall accuracy. This bias leads to **high accuracy on the majority class but poor performance on the minority class**, which is often the class of interest.

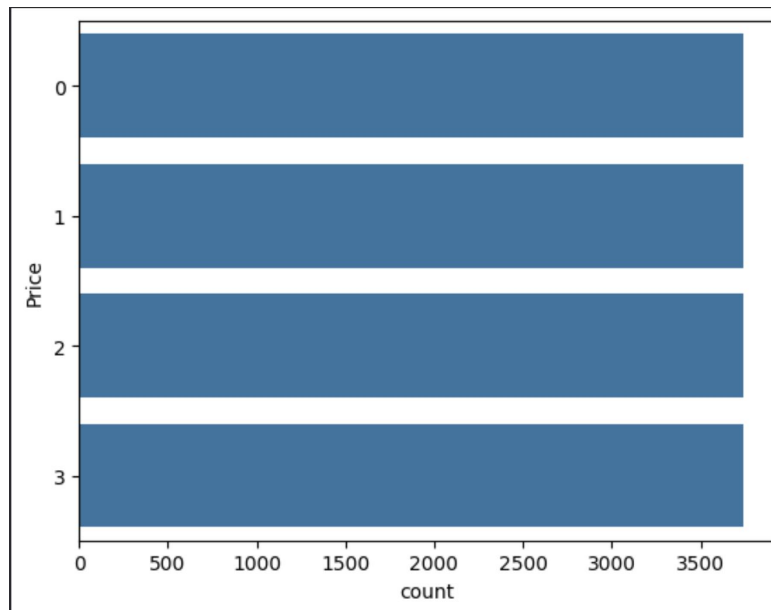
Handling Imbalanced Data

- ★ **Ensuring fairness and ethical considerations:** Imbalanced data can lead to models that perpetuate or amplify societal biases. Addressing class imbalance ensures that the model treats all classes fairly and avoids discriminatory outcomes.

Techniques to Handle Imbalanced Data

- ★ **Oversampling:** This technique involves increasing the number of instances in the minority class to balance the class distribution. Oversampling can be performed by duplicating existing minority instances or generating synthetic instances.
 - **Random oversampling:** Randomly duplicating minority class instances until the desired class balance is achieved.

Techniques to Handle Imbalanced Data



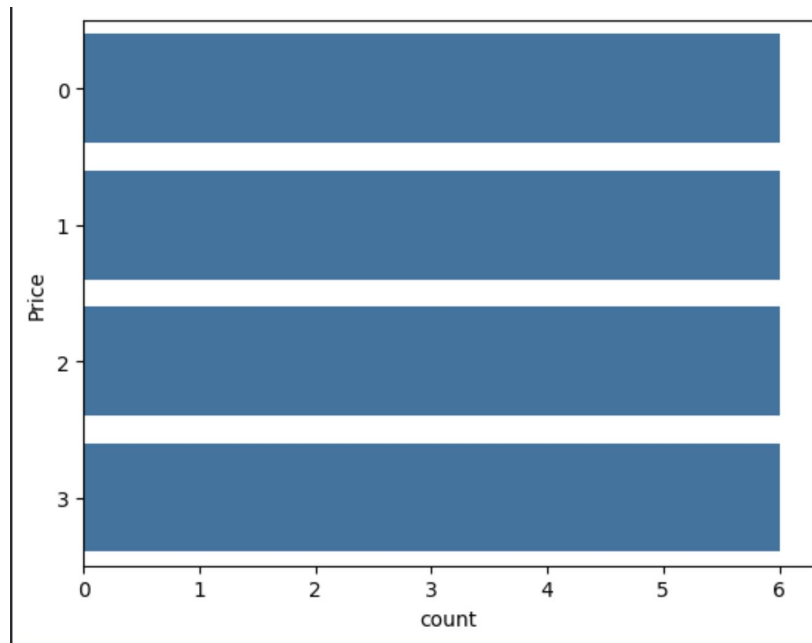
Techniques to Handle Imbalanced Data

- **Synthetic Minority Over-sampling Technique (SMOTE):**
Generating synthetic minority instances by interpolating between existing instances in the feature space.

Techniques to Handle Imbalanced Data

- ★ **Undersampling:** This technique involves **reducing the number of instances in the majority class to balance the class distribution**. Undersampling can be performed by randomly removing majority instances or using targeted approaches.
 - **Random undersampling:** Randomly removing majority class instances until the desired class balance is achieved.

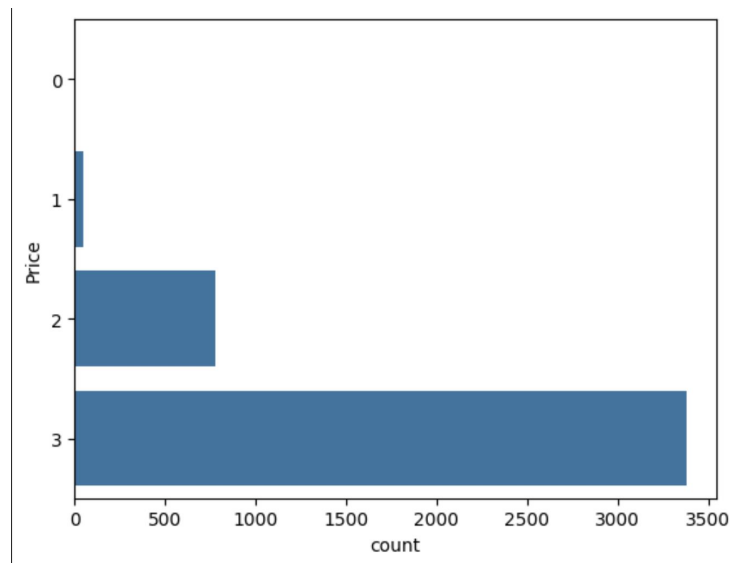
Techniques to Handle Imbalanced Data



Techniques to Handle Imbalanced Data

- ★ **Tomek links:** Identifying and removing majority instances that are close to minority instances, thereby reducing class overlap.

Techniques to Handle Imbalanced Data



Preprocessing Pipelines

- ★ Preprocessing pipelines are a way to **streamline and automate the various preprocessing steps involved in a machine learning workflow**. By encapsulating the preprocessing steps into a pipeline, data scientists can ensure consistency, reproducibility, and efficiency in their preprocessing tasks.

Preprocessing Pipelines

- ★ Building preprocessing pipelines with scikit-learn
 - **Scikit-learn**, a popular Python library for machine learning, provides the **Pipeline** and **ColumnTransformer** classes for building preprocessing pipelines.
 - The Pipeline class allows for **chaining multiple preprocessing steps and estimators together**. Each step in the pipeline is a tuple containing a string name and an estimator object. The output of each step is passed as input to the next step, creating a sequential workflow.

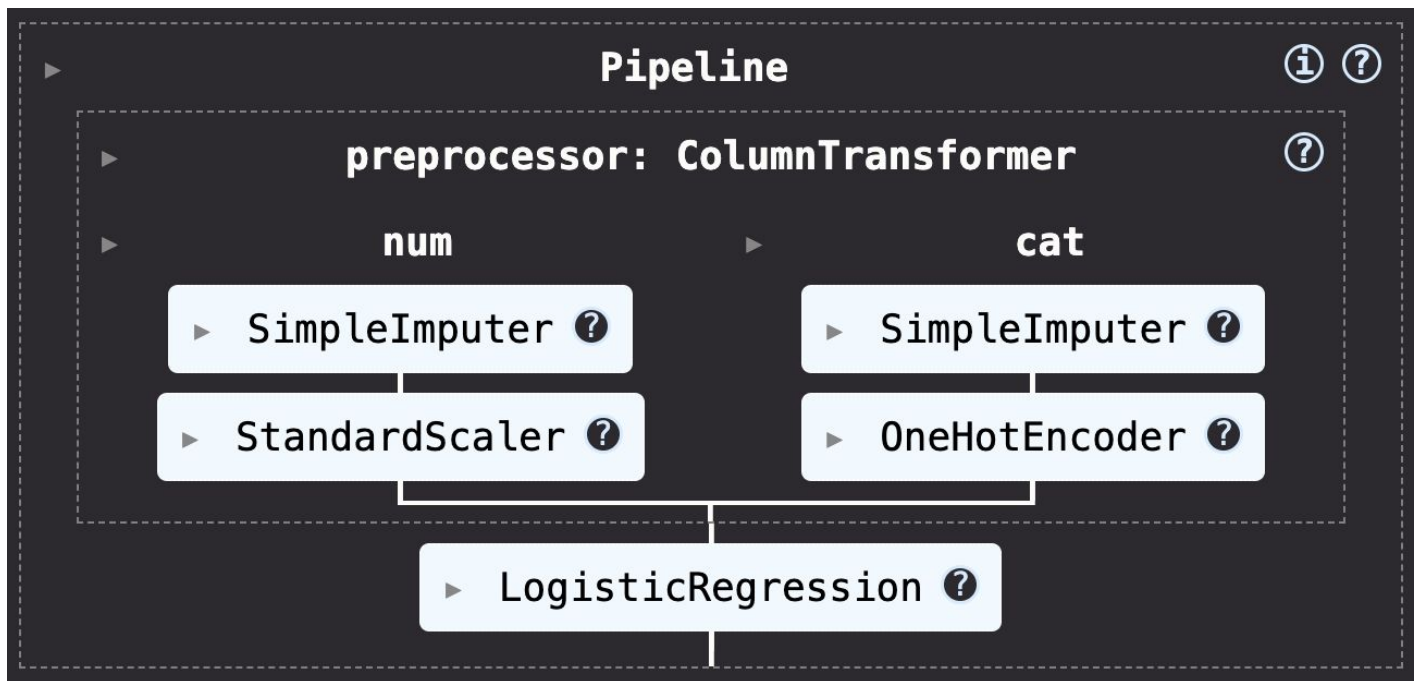
Preprocessing Pipelines

```
# Define the preprocessing steps
num_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler())
])

cat_pipeline = Pipeline([
    ('imputer', SimpleImputer(strategy='most_frequent')),
    ('onehot', OneHotEncoder(handle_unknown='ignore'))
])

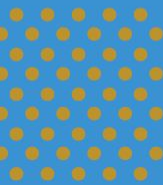
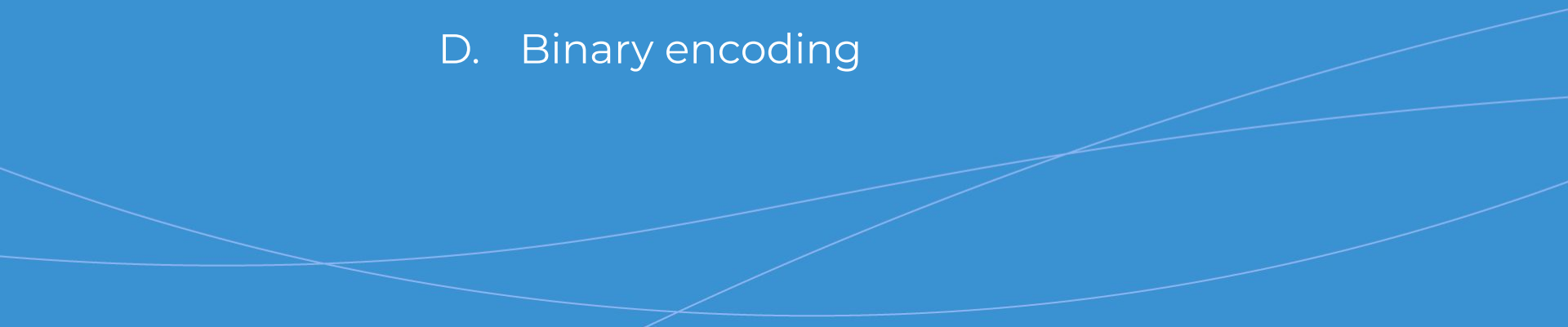
preprocessor = ColumnTransformer([
    ('num', num_pipeline, ['Age', 'Fare']),
    ('cat', cat_pipeline, ['Sex', 'Embarked'])
])
```


Preprocessing Pipelines




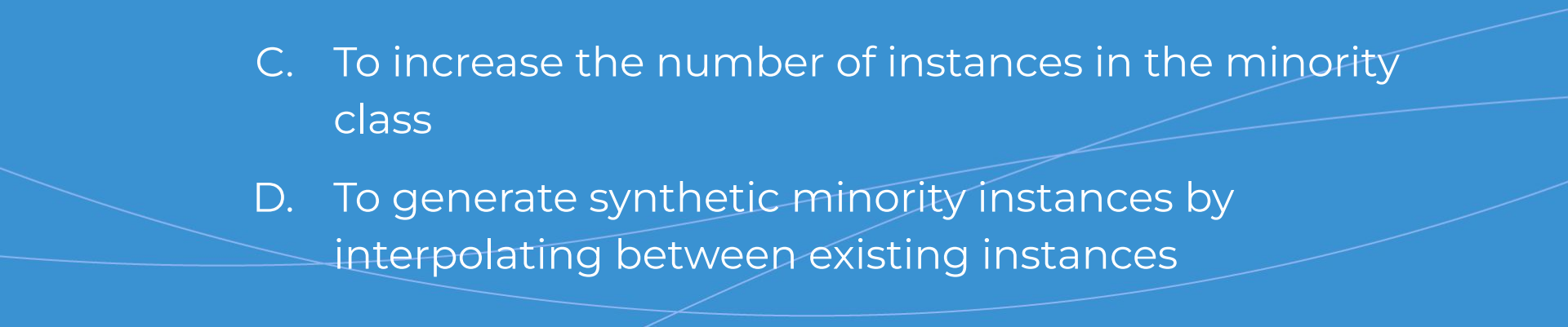


Which technique is used to generate higher-order terms from existing features?

- 
- A. Interaction features
 - B. Domain-specific features
 - C. Polynomial features
 - D. Binary encoding
- 


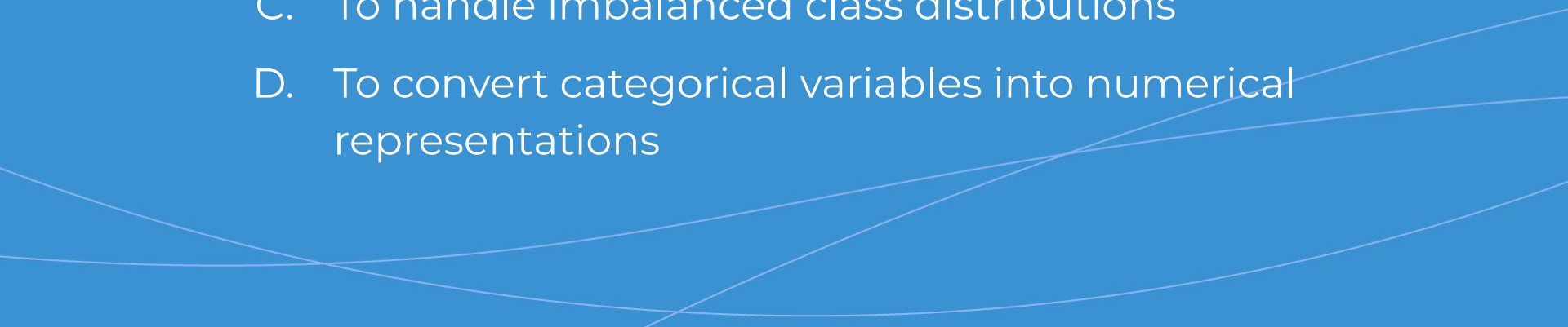


What is the purpose of oversampling in handling imbalanced data?

- 
- A. To reduce the number of instances in the majority class
 - B. To remove majority instances that are close to minority instances
 - C. To increase the number of instances in the minority class
 - D. To generate synthetic minority instances by interpolating between existing instances
- 



What is the benefit of using preprocessing pipelines?

- 
- A. To ensure consistency, reproducibility, and efficiency in preprocessing tasks
 - B. To create new features from existing data
 - C. To handle imbalanced class distributions
 - D. To convert categorical variables into numerical representations
- 

CoGrammar

Q & A SECTION

**Please use this time to ask
any questions relating to the
topic, should you have any.**



CoGrammar

Thank you for joining!