ARTISANAL E COMMERCE PLATFORM ON IBM CLOUD FOUNDRY

Introduction:

This involves identifying and correcting errors or inconsistencies in the data, such as missing values, outliers, and duplicates. Various techniques can be used for data cleaning, such as imputation, removal, and transformation.

Data pre-processing transforms the data into a format that is more easily and effectively processed in data mining, machine learning and other data science tasks. The techniques are generally used at the earliest stages of the machine learning and AI development pipeline to ensure accurate results.

1. **Dataset Selection:**

Choose the dataset that aligns with the objectives of your project. It should contain relevant information for the e-commerce application you're working on.

2. **Loading the Dataset :**

Use appropriate tools or libraries (e.g., Pandas for Python) to load the dataset into your development environment.

3. **Exploratory Data Analysis (EDA) :**

Perform preliminary data exploration to understand the structure of the dataset. This involves looking at features, data types, missing values, and basic statistics.

4. **Data Cleaning :**

Handle missing values, outliers, and any inconsistencies in the dataset. This may involve techniques like imputation, removal, or interpolation.

5. **Feature Engineering :**

Create new features or transform existing ones to extract more meaningful information. For an e-commerce application, this could involve things like creating customer segments, extracting product attributes, etc.

6. **Data Transformation** :

Convert categorical variables into numerical representations (e.g., one-hot encoding), normalize or standardize numerical features, and apply any other necessary transformations.

7**. Data Splitting** :

Divide the dataset into training, validation, and test sets. This is crucial for evaluating the performance of your models.

8**. Data Serialization (Optional):**

Depending on the size of your dataset, you might consider saving it in a serialized format (like CSV, HDF5, or Parquet) for faster loading in future sessions.

9.Loading the Data :

- **Selecting the Data Source** :

Identify where your data is coming from. It could be stored in various formats like CSV files, Excel spreadsheets, databases, JSON files, or even gathered from APIs.

- **Using Libraries or Tools :**

  The pending on the format of your data, you'll use specific libraries or tools to load it. For example, in Python, you might use pandas for tabular data (CSV, Excel) or libraries like requests for fetching data from APIs.

- **Understanding Data Structure :**

  Once loaded, it’s important to understand the structure of the data. This includes the number of columns, their names, and the type of data they hold (e.g., numerical, categorical, text).

10.Data Pre-processing :

- **Handling Missing Values :**

Check for and handle any missing values in the dataset. This could involve techniques like imputation (replacing missing values with estimates) or removal of rows or columns with too many missing values.

- **Dealing with Outliers** :

Identify and decide how to handle outliers in the data. Depending on the context, outliers might be removed, transformed, or left as is.

- **Encoding Categorical Data** :

If your dataset contains categorical variables (like colours, categories), you may need to encode them numerically using techniques like one-hot encoding or label encoding.

- **Feature Scaling/Normalization** :
  Standardize the scale of numerical features. This ensures that features with different scales don’t dominate the learning process. Techniques like Min-Max scaling or Standardization (Z-score scaling) can be used.

- **Feature Engineering** :
  This involves creating new features from existing ones or transforming features to better represent the underlying patterns in the data.

- **Splitting Data :**
  Divide the dataset into training and testing sets. This helps in evaluating the model’s performance on unseen data.

- **Shuffling Data (for ML)** :
  If applicable, randomly shuffle the data to ensure that the model doesn’t learn any spurious patterns related to the order of the samples.

- **Data Augmentation (for Image Data) :**

  In tasks involving images, data augmentation techniques can be applied to increase the diversity of training samples.

- **Text Data Pre-processing (for NLP) :**

  This can involve tasks like tokenization, stop word removal, stemming/lemmatization, and converting text to numerical representations.

- **Handling Time Series Data** :

  For time series data, special techniques like lagging, rolling statistics, or seasonal decomposition may be applied.

- **Saving Pre-processed Data** :

  It's often a good idea to save the pre-processed data to avoid repeating these steps every time you work on the project.

- **To load and pre-process a dataset in your code, you'll typically follow these steps:**

1. **Import Necessary Libraries :**

- Depending on your programming language, you'll import libraries that will help with data handling and pre-processing. For example, in Python, you might use libraries like pandas for data handling and scikit-learn for pre-processing tasks.

2. **Load the Dataset :**

- Use the appropriate method from your chosen library to load the dataset. For example, in Python with pandas, you might use `pd.read csv()` for a CSV file.

Example in Python :

python Code

```
Import pandas as pd

# Load dataset
Df = pd.read_csv('your_dataset.csv')
```

3. **Explore the Dataset**:

- Print out some basic information about the dataset like the first few rows, column names, data types, and summary statistics.

Example in Python:

python code

```
# Print first few rows
Print(df.head())

# Get column names
Print(df.columns)

# Get data types
Print(df.dtypes)

# Get summary statistics
Print(df.describe())
```

4. **Handle Missing Values:**

- Identify and handle any missing values. This could involve techniques like filling missing values with averages or removing rows/columns with too many missing values.

Example in Python:

python code

```
# Check for missing values
Print(df.isnull().sum())

# Handle missing values (for example, fill with mean)
Df.fillna(df.mean(), inplace=True)
```

5. **Preprocess the Data** :

- Depending on the nature of your data, apply specific preprocessing techniques. This could include encoding categorical variables, scaling numerical features, and more.

Example in Python:

python code

```
From sklearn.preprocessing import StandardScaler, LabelEncoder
```

```
  # Example: Scaling numerical features
  Scaler = StandardScaler()
  Df[[‘numerical_column’]] =
scaler.fit_transform(df[[‘numerical_column’]])

  # Example: Encoding categorical variables
  Le = LabelEncoder()
  Df[‘categorical_column’] =
le.fit_transform(df[‘categorical_column’])
```

6. **Split the Data (if applicable):**

  - If you’re working on a machine learning project, split the dataset into training and testing sets to evaluate the model.

  Example in Python:

  python code

```
  From sklearn.model_selection import train_test_split
```

```
   X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)
```

7. **Save Preprocessed Data(optional):**

- You can save the pre-processed data to avoid repeating these steps in the future.

Example in Python:

python code

```
Df.to_csv('preprocessed_data.csv', index=False)
```

CONCLUSION:

Data pre-processing is an essential step in the data mining process and plays a crucial role in ensuring that the data is in a suitable format for analysis. This article provides a comprehensive guide to data pre-processing techniques, including data cleaning, integration, reduction, and transformation.