

Unit-4

Faculty: D Sai Kumar

Dept. of CSE, UCE, OU

Data Cleaning

- Data analysts and scientists spend most of their time cleaning data and pre-processing messy datasets.
- Data cleaning and preprocessing is the process of identifying, updating, and removing corrupt or incorrect data.
- Cleaning and pre-processing results in high-quality data for robust and error-free analysis.
- Quality data can beat complex algorithms and outperform simple and less complex algorithms. In this context, high quality means accurate, complete, and consistent data.
- Data cleaning is a set of activities such as handling missing values, removing outliers, feature encoding, scaling, transformation, and splitting.

Exploring data

- We will explore data by performing **Exploratory Data Analysis (EDA)**.
- EDA is the most critical and most important component of the data analysis process.
- EDA offers the following benefits:
 1. It provides an initial glimpse of data and its context.
 2. It captures quick insights and identifies the potential drivers from the data for predictive analysis. It finds the queries and questions that can be answered for decision-making purposes.
 3. It assesses the quality of the data and helps us build the road map for data cleaning and preprocessing.
 4. It finds missing values, outliers, and the importance of features for analysis.
 5. EDA uses descriptive statistics and visualization techniques to explore data.

- In EDA, the first step is to read the dataset. We can read the dataset using pandas.
- After reading the data, we can explore the data.
- This initial exploration will help us understand the data and gain some domain insights.

```
# import pandas
```

```
import pandas as pd
```

```
# Read the data using csv
```

```
data=pd.read_csv('employee.csv')
```

```
# See initial 5 records
```

```
data.head()
```

	name	age	income	gender	department	grade	performance_score
0	Allen Smith	45.0	NaN	NaN	Operations	G3	723
1	S Kumar	NaN	16000.0	F	Finance	G0	520
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674
3	Ying Chin	45.0	65000.0	F	Sales	G3	556
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2	711

```
# See last 5 records
```

```
data.tail()
```

	name	age	income	gender	department	grade	performance_score
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2	711
5	Satyam Sharma	NaN	62000.0	NaN	Sales	G3	649
6	James Authur	54.0	NaN	F	Operations	G3	53
7	Josh Wills	54.0	52000.0	F	Finance	G3	901
8	Leo Duck	23.0	98000.0	M	Sales	G4	709

```
# Print list of columns in the data
```

```
print(data.columns)
```

Output:

```
Index(['name', 'age', 'income', 'gender', 'department', 'grade', 'performance_score'], dtype='object')
```

Print the shape of a DataFrame

```
print(data.shape)
```

Output:

(9, 7)

Check the information of DataFrame

```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9 entries, 0 to 8
Data columns (total 7 columns):
name                9 non-null object
age                 7 non-null float64
income              7 non-null float64
gender              7 non-null object
department          9 non-null object
grade               9 non-null object
performance_score    9 non-null int64
dtypes: float64(2), int64(1), object(4)
memory usage: 584.0+ bytes
```

Check the descriptive statistics

data.describe()

	age	income	performance_score
count	7.000000	7.000000	9.000000
mean	40.428571	52857.142857	610.666667
std	12.204605	26028.372797	235.671912
min	23.000000	16000.000000	53.000000
25%	31.000000	38500.000000	556.000000
50%	45.000000	52000.000000	674.000000
75%	49.500000	63500.000000	711.000000
max	54.000000	98000.000000	901.000000

Filtering data to weed out the noise

- In the last two decades, the data size of companies and government agencies has increased due to digitalization.
- This also caused an increase in consistency, errors, and missing values.
- Data filtering is responsible for handling such issues and optimizing them for management, reporting, and predictions.
- The filtering process boosts the accuracy, relevance, completeness, consistency, and quality of the data by processing dirty, messy, or coarse datasets.
- It is a very crucial step for any kind of data management because it can make or break a competitive edge of business.
- Data scientists need to master the skill of data filtering.
- Different kinds of data need different kinds of treatment. That's why a systematic approach to data filtering needs to be taken.

Column-wise filtration

- Data can be filtered either column-wise or row-wise.
- We can filter columns using the filter() method.
- The slicing []. filter() method selects the columns when they're passed as a list of columns.

Filter columns

```
data.filter(['name', 'department'])
```

	name	department
0	Allen Smith	Operations
1	S Kumar	Finance
2	Jack Morgan	Finance
3	Ying Chin	Sales
4	Dheeraj Patel	Operations
5	Satyam Sharma	Sales
6	James Authur	Operations
7	Josh Wills	Finance
8	Leo Duck	Sales

- we can also filter columns using slicing. In slicing, a single column does not need a list, but when we are filtering multiple columns, then they should be on the list.
- The output of a single column is a pandas Series.
- If we want the output as a DataFrame, then we need to put the name of the single column into a list.

```
# Filter column "name"
data['name']
```

Output:

```
0    Allen Smith
1     S Kumar
2   Jack Morgan
3    Ying Chin
4   Dheeraj Patel
5   Satyam Sharma
6   James Authur
7    Josh Wills
8     Leo Duck
```

Name: name, dtype: object

```
# Filter column "name"
data[['name']]
```

	name
0	Allen Smith
1	S Kumar
2	Jack Morgan
3	Ying Chin
4	Dheeraj Patel
5	Satyam Sharma
6	James Authur
7	Josh Wills
8	Leo Duck

Filter two columns: name and department

data[['name','department']]

	name	department
0	Allen Smith	Operations
1	S Kumar	Finance
2	Jack Morgan	Finance
3	Ying Chin	Sales
4	Dheeraj Patel	Operations
5	Satyam Sharma	Sales
6	James Authur	Operations
7	Josh Wills	Finance
8	Leo Duck	Sales

row-wise filtration

- We can filter data using indices, slices, and conditions.
- In indices, we have to pass the index of the record, while for slicing, we need to pass the slicing range.

Select rows for the specific index

```
data.filter([0,1,2],axis=0)
```

	name	age	income	gender	department	grade	performance_score
0	Allen Smith	45.0	NaN	NaN	Operations	G3	723
1	S Kumar	NaN	16000.0	F	Finance	G0	520
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674

Filter data using slicing

```
data[2:5]
```

	name	age	income	gender	department	grade	performance_score
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674
3	Ying Chin	45.0	65000.0	F	Sales	G3	556
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2	711

- In condition-based filtration, we have to pass some conditions in square brackets, [], or brackets, ().
- For a single value, we use the == (double equal to) condition, while for multiple values, we use the isin() function and pass the list of values.

Filter data for specific value

```
data[data.department=='Sales']
```

	name	age	income	gender	department	grade	performance_score
3	Ying Chin	45.0	65000.0	F	Sales	G3	556
5	Satyam Sharma	NaN	62000.0	NaN	Sales	G3	649
8	Leo Duck	23.0	98000.0	M	Sales	G4	709

Select data for multiple values

```
data[data.department.isin(['Sales','Finance'])]
```

	name	age	income	gender	department	grade	performance_score
1	S Kumar	NaN	16000.0	F	Finance	G0	520
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674
3	Ying Chin	45.0	65000.0	F	Sales	G3	556
5	Satyam Sharma	NaN	62000.0	NaN	Sales	G3	649
7	Josh Wills	54.0	52000.0	F	Finance	G3	901
8	Leo Duck	23.0	98000.0	M	Sales	G4	709

Filter employee who has more than 700 performance score

```
data[(data.performance_score >=700)]
```

	name	age	income	gender	department	grade	performance_score
0	Allen Smith	45.0	NaN	NaN	Operations	G3	723
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2	711
7	Josh Wills	54.0	52000.0	F	Finance	G3	901
8	Leo Duck	23.0	98000.0	M	Sales	G4	709

Filter employee who has more than 500 and less than 700 performance score

```
data[(data.performance_score >=500) & (data.performance_score < 700)]
```

	name	age	income	gender	department	grade	performance_score
1	S Kumar	NaN	16000.0	F	Finance	G0	520
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674
3	Ying Chin	45.0	65000.0	F	Sales	G3	556
5	Satyam Sharma	NaN	62000.0	NaN	Sales	G3	649

We can also try the query() method. This method queries the columns using a Boolean expression.

```
# Filter employee who has performance score of less than 500
```

```
data.query('performance_score<500')
```

	name	age	income	gender	department	grade	performance_score
6	James Authur	54.0	NaN	F	Operations	G3	53

Handling missing values

- Missing values are the values that are absent from the data. Absent values can occur due to human error, privacy concerns, or the value not being filled in by the respondent filling in the survey. This is the most common problem in data science and the first step of data preprocessing.
- Missing values affect a machine learning model's performance.
- Missing values can be handled in the following ways:
 1. Drop the missing value records.
 2. Fill in the missing value manually.
 3. Fill in the missing values using the measures of central tendency, such as mean, median, and mode. The mean is used to impute the numeric feature, the median is used to impute the ordinal feature, and the mode or highest occurring value is used to impute the categorical feature.
 4. Fill in the most probable value using machine learning models such as regression, decision trees, KNNs.
- It is important to understand that in some cases, missing values will not impact the data.
- For example, driving license numbers, social security numbers, or any other unique identification numbers will not impact the machine learning models because they can't be used as features in the model.

Dropping missing values

- In Python, missing values can be dropped using the dropna() function. dropna takes one argument: how.
- how can take two values: all or any. any drops certain rows that contain NAN or missing values, while all drops all the rows contains NAN or missing values:

Drop missing value rows using dropna() function

Read the data

```
data=pd.read_csv('employee.csv')
```

```
data=data.dropna()
```

```
data
```

	name	age	income	gender	department	grade	performance_score
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674
3	Ying Chin	45.0	65000.0	F	Sales	G3	556
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2	711
7	Josh Wills	54.0	52000.0	F	Finance	G3	901
8	Leo Duck	23.0	98000.0	M	Sales	G4	709

Filling in a missing value

- In Python, missing values can be dropped using the `fillna()` function. The `fillna()` function takes one value that we want to fill at the missing place. We can fill in the missing values using the mean, median, and mode:

```
# Read the data
```

```
data=pd.read_csv('employee.csv')
```

```
# Fill all the missing values in the age column with mean of the age column
```

```
data['age']=data.age.fillna(data.age.mean())
```

```
data
```

	name	age	income	gender	department	grade	performance_score
0	Allen Smith	45.000000	NaN	NaN	Operations	G3	723
1	S Kumar	40.428571	16000.0	F	Finance	G0	520
2	Jack Morgan	32.000000	35000.0	M	Finance	G2	674
3	Ying Chin	45.000000	65000.0	F	Sales	G3	556
4	Dheeraj Patel	30.000000	42000.0	F	Operations	G2	711
5	Satyam Sharma	40.428571	62000.0	NaN	Sales	G3	649
6	James Authur	54.000000	NaN	F	Operations	G3	53
7	Josh Wills	54.000000	52000.0	F	Finance	G3	901
8	Leo Duck	23.000000	98000.0	M	Sales	G4	709

- how to fill in the missing values using the median:

Fill all the missing values in the income column with a median of the income column

```
data['income']=data.income.fillna(data.income.median())
```

data

	name	age	income	gender	department	grade	performance_score
0	Allen Smith	45.000000	52000.0	NaN	Operations	G3	723
1	S Kumar	40.428571	16000.0	F	Finance	G0	520
2	Jack Morgan	32.000000	35000.0	M	Finance	G2	674
3	Ying Chin	45.000000	65000.0	F	Sales	G3	556
4	Dheeraj Patel	30.000000	42000.0	F	Operations	G2	711
5	Satyam Sharma	40.428571	62000.0	NaN	Sales	G3	649
6	James Authur	54.000000	52000.0	F	Operations	G3	53
7	Josh Wills	54.000000	52000.0	F	Finance	G3	901
8	Leo Duck	23.000000	98000.0	M	Sales	G4	709

- how to fill in missing values using the mode:

Fill all the missing values in the gender column(category column) with the mode of the gender column

```
data['gender']=data['gender'].fillna(data['gender'].mode()[0])
```

data

	name	age	income	gender	department	grade	performance_score
0	Allen Smith	45.000000	52000.0	F	Operations	G3	723
1	S Kumar	40.428571	16000.0	F	Finance	G0	520
2	Jack Morgan	32.000000	35000.0	M	Finance	G2	674
3	Ying Chin	45.000000	65000.0	F	Sales	G3	556
4	Dheeraj Patel	30.000000	42000.0	F	Operations	G2	711
5	Satyam Sharma	40.428571	62000.0	F	Sales	G3	649
6	James Authur	54.000000	52000.0	F	Operations	G3	53
7	Josh Wills	54.000000	52000.0	F	Finance	G3	901
8	Leo Duck	23.000000	98000.0	M	Sales	G4	709

Handling outliers

- Outliers are those data points that are distant from most of the similar points.
- Outliers cause problems when it comes to building predictive models, such as long model training times, poor accuracy, an increase in error variance, a decrease in normality, and a reduction in the power of statistical tests.
- There are two types of outliers: univariate and multivariate.
- Univariate outliers can be found in single variable distributions, while multivariates can be found in n-dimensional spaces.
- We can detect and handle outliers in the following ways:

Box Plot: We can use a box plot to create a bunch of data points through quartiles. It groups the data points between the first and third quartile into a rectangular box. The box plot also displays the outliers as individual points using the interquartile range.

Scatter Plot: A scatter plot displays the points (or two variables) on the two dimensional chart. One variable is placed on the x-axis, while the other is placed on the y-axis.

Z-Score: The Z-score is a kind of parametric approach to detecting outliers. It assumes a normal distribution of the data. The outlier lies in the tail of the normal curve distribution and is far from the mean:

$$Z = \frac{x - \mu}{\sigma}$$

Interquartile Range (IQR): IQR is a robust statistical measure of data dispersion. It is the difference between the third and first quartile. These quartiles can be visualized in a box plot. This is also known as the midspread, the middle 50%, or H-spread:

$$IQR = Q3 - Q1$$

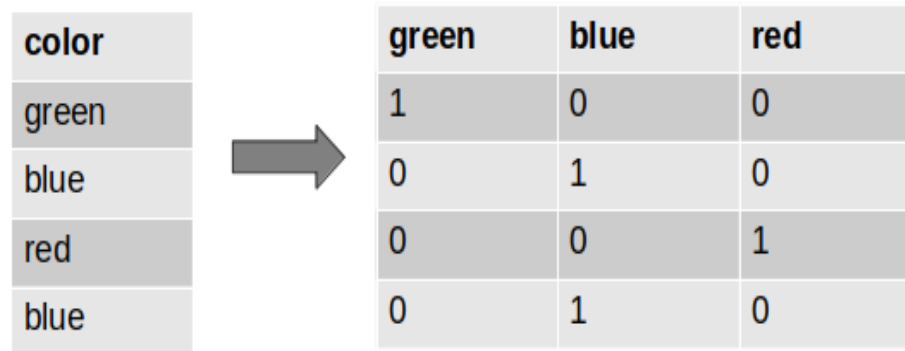
Percentile: A percentile is a statistical measure that divides data into 100 groups of equal size. Its value indicates the percentage of the population below that value. For example, the 95th percentile means 95% of people fall under this category.

Feature encoding techniques

- Machine learning models are mathematical models that required numeric and integer values for computation.
- Such models can't work on categorical features. That's why we often need to convert categorical features into numerical ones.
- Machine learning model performance is affected by what encoding technique we use. Categorical values range from 0 to N-1 categories.

one-hot encoding

- One-hot encoding transforms the categorical column into labels and splits the column into multiple columns.
- The numbers are replaced by binary values such as 1s or 0s.



The diagram illustrates the process of one-hot encoding. On the left, a single column labeled 'color' contains the values 'green', 'blue', 'red', and 'blue'. A large gray arrow points to the right, where a new table structure is shown. This new structure has three columns labeled 'green', 'blue', and 'red'. Each row in this new table corresponds to a row in the original 'color' column. The values are binary: 1 for the color that matches the column header, and 0 for the others. For example, the first row (green) has a 1 in the 'green' column and 0s in the 'blue' and 'red' columns.

color	green	blue	red
green	1	0	0
blue	0	1	0
red	0	0	1
blue	0	1	0

- One-hot encoding can also be performed using the `get_dummies()` function.

```
# Read the data
```

```
data=pd.read_csv('employee.csv')
```

```
# Dummy encoding
```

```
encoded_data = pd.get_dummies(data['gender'])
```

```
# Join the encoded _data with original dataframe
```

```
data = data.join(encoded_data)
```

	name	age	income	gender	department	grade	performance_score	F	M
0	Allen Smith	45.0	NaN	NaN	Operations	G3	723	0	0
1	S Kumar	NaN	16000.0	F	Finance	G0	520	1	0
2	Jack Morgan	32.0	35000.0	M	Finance	G2	674	0	1
3	Ying Chin	45.0	65000.0	F	Sales	G3	556	1	0
4	Dheeraj Patel	30.0	42000.0	F	Operations	G2	711	1	0

Label encoding

- Label encoding is also known as integer encoding. Integer encoding replaces categorical values with numeric values. Here, the unique values in variables are replaced with a sequence of integer values.
- For example, let's say there are three categories: red, green, and blue. These three categories were encoded with integer values; that is, red is 0, green is 1, and blue is 2.

ordinal encoder

- Ordinal encoding is similar to label encoding, except there's an order to the encoding.
- The output encoding will start from 0 and end at one less than the size of the categories.
- Let's look at an example containing employee grades such as G0, G1, G2, G3, and G4. These five grades have been encoded with ordinal integer values; that is, G0 is 0, G1 is 1, G2 is 2, G3 is 3, and G4 is 4.
- We can define the order of the values as a list and pass it to the category parameter.
- The ordinal encoder uses the integer or numeric values to encode.
- Here, the integer and numeric values are ordinal in nature. This encoding helps machine learning algorithms take advantage of this ordinal relationship.

Correlation: Introducing correlation

- Correlation is a statistical measure that describes the extent to which two variables change together.
- It quantifies the strength and direction of the linear relationship between them.
- **Positive Correlation:** As one variable increases, the other tends to increase.
- **Negative Correlation:** As one variable increases, the other tends to decrease.
- **No Correlation:** There is no clear linear relationship between the variables.

Types of analysis

Univariate Analysis:

- This type of analysis examines a single variable at a time.
- Its purpose is to describe the distribution of the variable, identify central tendencies (mean, median, mode), and understand its spread (variance, standard deviation, range).

Examples:

- Analyzing the distribution of passenger ages on the Titanic.
- Examining the frequency of different passenger classes.

Bivariate Analysis:

- This type of analysis explores the relationship between two variables.
- It aims to determine whether there is an association between the variables and, if so, to quantify the strength and direction of that association.

Examples:

- Investigating the relationship between passenger age and fare paid.
- Examining the correlation between passenger class and survival rate.

Multivariate Analysis:

- This type of analysis examines the relationships among three or more variables simultaneously.
- It allows for a more comprehensive understanding of complex interactions and dependencies.

Examples:

- Modeling survival rate as a function of passenger class, sex, and age.
- Analyzing the combined effects of multiple factors on fare paid.

Discussing multivariate analysis using the Titanic dataset

The Titanic dataset is a classic example for multivariate analysis. Here's how we can approach it:

1. Feature Selection and Engineering:

- Identify relevant features (e.g., Pclass, Sex, Age, Fare, SibSp, Parch).
- Create new features (e.g., family size, age groups).

2. Visualizations:

- Use pair plots to visualize relationships between numerical features.
- Create 3D scatter plots to examine relationships between three variables.
- Use heatmaps to visualize correlation matrices.
- Use stacked bar charts to see survival rates across multiple categorical variables.

3. Statistical Analysis:

- Use ANOVA to analyze the impact of categorical variables on numerical variables.
- Perform chi-square tests to analyze relationships between categorical variables.
- Apply logistic regression to model survival probability.

4. Machine Learning:

- Train classification models (e.g., Random Forest, Support Vector Machines) to predict survival.
- Perform feature importance analysis to identify key predictors.
- Evaluate Model performance.