UNIT 2

Data Collection and Preprocessing

Data Acquisition Methods and Sources

Data acquisition (also called data mining) is the process of gathering data.

Sometimes data is gathered before we know what to do with it. When that happens, it is important to take a step back and define what questions can be answered with the available data.

In addition, some things to consider when acquiring data are:

- What data is needed to achieve the goal?
- How much data is needed?
- Where and how can this data be found?
- What legal and privacy concerns should be considered?

The role of data collection: Example

Imagine for a moment that you are collecting data about books. You decided to record the title, author, and number of pages of all the books in your local library. You decided not to include language, subtitles, editors, or publishers.

If you want to publish this data to make it available to others, you would need to document how you measured your variables (i.e., were appendices included in the page count?) and the parameters for collection (i.e., your local library). This is your methodology.

How the data was collected (the methodology) directly affects the questions we can ask and what generalizations we can make.

Data sources

Data can be acquired from many different sources. Broadly, they can be categorized into **primary data** and **secondary data**.

Primary data is data collected by the individual or organization who will be doing the analysis. Examples include:

- Experiments (e.g., wet lab experiments like gene sequencing)
- Observations (e.g., surveys, sensors, *in situ* collection)
- Simulations (e.g., theoretical models like climate models)
- Scraping or compiling (e.g., webscraping, text mining)

Secondary data is data collected by someone else and is typically published for public use. Examples include:

- Any primary data that was collected by someone else
- Institutionalized data banks (e.g., census, gene sequences)

[Cleaned Vs Raw Data]

Data file formats:

Data can come in a variety of different file formats, depending on the type of data.

Being able to open and convert between these file types opens a whole world of data that is otherwise inaccessible. Examples of file formats include:

- Tabular (e.g., **.csv**, **.tsv**, **.xlsx**)
- Non-tabular (e.g., **.txt**, **.rtf**, **.xml**)
- Image (e.g., **.png**, **.jpg**, **.tif**)
- Agnostic (e.g., **.dat**)

Where to get data:

Primary data

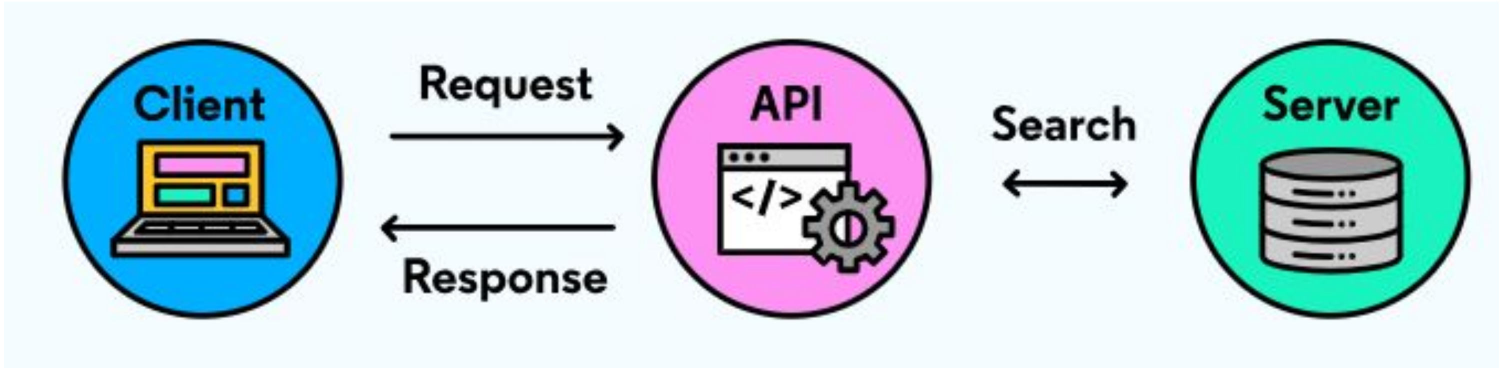
surveys and simulations are common methods for acquiring primary data.
Web Scraping is also a special case of primary data collection by extracting or copying data directly from a website.

Secondary data

Secondary data can be obtained from many different websites.Each repository or individual dataset has its own terms of use and method for downloading. Some of the most popular repositories include:

- Kaggle
- GitHub
- KDnuggets
- UCI Machine Learning Repository
- US Government’s Open Data
- Five Thirty Eight
- Amazon Web Services
- BuzzFeed
- Data is Plural
- Harvard HCI

Secondary data can sometimes be obtained via an application programming interface (API). APIs are built around the **HTTP request/response cycle**. A client (you) sends a request for data to a website's server through an API call. Then, the server searches its database and responds either with the data, or an error stating that the request cannot be fulfilled.



Exploratory data analysis

A method used to analyze and summarize data sets.

Exploratory data analysis (EDA) is used by data scientists to analyze and investigate data sets and summarize their main characteristics, often employing data visualization methods.

It can help identify obvious errors, as well as better understand patterns within the data, detect outliers or anomalous events, find interesting relations among the variables.

Goals of EDA

1. Data Cleaning: EDA involves examining the information for errors, lacking values, and inconsistencies. It includes techniques including records imputation, managing missing statistics, and figuring out and getting rid of outliers.

2. Descriptive Statistics: EDA utilizes precise records to recognize the important tendency, variability, and distribution of variables. Measures like suggest, median, mode, preferred deviation, range, and percentiles are usually used.

3. Data Visualization: EDA employs visual techniques to represent the statistics graphically. Visualizations consisting of histograms, box plots, scatter plots, line plots, heatmaps, and bar charts assist in identifying styles, trends, and relationships within the facts.

4. Feature Engineering: EDA allows for the exploration of various variables and their adjustments to create new functions or derive meaningful insights. Feature engineering can contain scaling, normalization, binning, encoding express variables, and creating interplay or derived variables.

5. Correlation and Relationships: EDA allows discover relationships and dependencies between variables. Techniques such as correlation analysis, scatter plots, and pass-tabulations offer insights into the power and direction of relationships between variables.

6. Data Segmentation: EDA can contain dividing the information into significant segments based totally on sure standards or traits. This segmentation allows advantage insights into unique subgroups inside the information and might cause extra focused analysis.

7. Hypothesis Generation: EDA aids in generating hypotheses or studies questions based totally on the preliminary exploration of the data. It facilitates form the inspiration for in addition evaluation and model building.

8. Data Quality Assessment: EDA permits for assessing the nice and reliability of the information. It involves checking for records integrity, consistency, and accuracy to make certain the information is suitable for analysis.

Types of EDA

Depending on the number of columns we are analyzing we can divide EDA into:

1. Univariate Analysis: This is simplest form of data analysis, where the data being analyzed consists of just one variable. Since it’s a single variable, it doesn’t deal with causes or relationships. The main purpose of univariate analysis is to describe the data and find patterns that exist within it.

2. Bivariate Analysis: Bivariate evaluation involves exploring the connection between variables. It enables find associations, correlations, and dependencies between pairs of variables. Scatter plots, line plots, correlation matrices, and move-tabulation are generally used strategies in bivariate analysis.

3. Multivariate Analysis: Multivariate analysis extends bivariate evaluation to encompass greater than variables. It ambitions to apprehend the complex interactions and dependencies among more than one variables in a records set. Techniques inclusive of heatmaps, parallel coordinates, aspect analysis, and principal component analysis (PCA) are used for multivariate analysis.

4. Time Series Analysis: This type of analysis is mainly applied to statistics sets that have a temporal component. Time collection evaluation entails inspecting and modeling styles, traits, and seasonality inside the statistics through the years. Techniques like line plots, autocorrelation analysis, transferring averages, and ARIMA (AutoRegressive Integrated Moving Average) fashions are generally utilized in time series analysis.

5. Missing Data Analysis: Missing information is a not unusual issue in datasets, and it may impact the reliability and validity of the evaluation. Missing statistics analysis includes figuring out missing values, know-how the patterns of missingness, and using suitable techniques to deal with missing data. Techniques along with lacking facts styles, imputation strategies, and sensitivity evaluation are employed in lacking facts evaluation.

6. Outlier Analysis: Outliers are statistics factors that drastically deviate from the general sample of the facts. Outlier analysis includes identifying and knowledge the presence of outliers, their capability reasons, and their impact at the analysis. Techniques along with box plots, scatter plots, z-rankings, and clustering algorithms are used for outlier evaluation.

7. Data Visualization: Data visualization is a critical factor of EDA that entails creating visible representations of the statistics to facilitate understanding and exploration. Various visualization techniques, inclusive of bar charts, histograms, scatter plots, line plots, heatmaps, and interactive dashboards, are used to represent exclusive kinds of statistics.

Exploratory Data Analysis (EDA) Using Python Libraries

Dataset: employees.csv

contains 8 columns namely – First Name, Gender, Start Date, Last Login, Salary, Bonus%, Senior Management, and Team.

read the dataset

```
import pandas as pd

import numpy as np

# read dataset using pandas

df = pd.read_csv('employees.csv')

df.head()

df.shape                        #  shape of the data

# The describe() function applies basic statistical computations on the dataset like extreme values, count
of data points standard deviation, etc.

df.describe()
```

Note we can also get the description of categorical columns of the dataset if we specify ***include ='all'*** in the describe function.

```
#the columns and their data types

df.info()

# Changing Dtype from Object to Datetime

# convert "Start Date" column to datetime data type

df['Start Date'] = pd.to_datetime(df['Start Date'])
```

#the number of unique elements

This will help us in deciding which type of encoding to choose for converting categorical columns into numerical columns.

```
df.nunique()
```

Handling Missing Values

It can occur when no information is provided for one or more items or for a whole unit.

For Example, Suppose different users being surveyed may choose not to share their income, and some users may choose not to share their address in this way many datasets went missing.

Missing Data can also refer to as NA(Not Available) values in pandas.

There are several useful functions for detecting, removing, and replacing null values in Pandas DataFrame :

- isnull()
- notnull()
- dropna()
- fillna()
- replace()
- interpolate()

```
df.isnull().sum()                  #check if there are any missing values in our dataset or
not
```

For handling the missing values there can be several cases like dropping the rows containing NaN or replacing NaN with either mean, median, mode, or some other value.

Now, let’s try to fill in the missing values of gender with the string “No Gender”.

```
df["Gender"].fillna("No Gender", inplace = True)

df.isnull().sum()
```

```
#fill the senior management with the mode value.

mode = df['Senior Management'].mode().values[0]

df['Senior Management']= df['Senior Management'].replace(np.nan, mode)

df.isnull().sum()

# for the first name and team, we cannot fill the missing values with arbitrary data,
# so, let’s drop all the rows containing these missing values.

df = df.dropna(axis = 0, how ='any')

print(df.isnull().sum())

df.shape
```

Data Encoding: There are some models like Linear Regression which does not work with categorical dataset in that case we should try to encode categorical dataset into the numerical column. we can use different methods for encoding like Label encoding or One-hot encoding.

```
from sklearn.preprocessing import LabelEncoder

# create an instance of LabelEncoder

le = LabelEncoder()


# fit and transform the "Gender" column with LabelEncoder

df['Gender'] = le.fit_transform (df['Gender'])
```

Data visualization

Data Visualization is the process of analyzing data in the form of graphs or maps, making it a lot easier to understand the trends or patterns in the data.

Histogram

It can be used for both uni and bivariate analysis.

```
# importing packages

import seaborn as sns

import matplotlib.pyplot as plt

sns.histplot(x='Salary', data=df, )

plt.show()
```

Boxplot : It can also be used for univariate and bivariate analyses.

```
# importing packages
import seaborn as sns
import matplotlib.pyplot as plt
sns.boxplot( x="Salary", y='Team', data=df, )
plt.show()
```

For multivariate analysis, we can use pairplot()method of the seaborn module. We can also use it for the multiple pairwise bivariate distributions in a dataset.

```
# importing packages

import seaborn as sns

import matplotlib.pyplot as plt

sns.pairplot(df, hue='Gender', height=2)
```

Handling Outliers

Outlier is an observation in a given dataset that lies far from the rest of the observations.

That means an outlier is vastly larger or smaller than the remaining values in the set.

An outlier may occur due to the variability in the data, or due to experimental error/human error.

They may indicate an experimental error or heavy skewness in the data(heavy-tailed distribution).

What Do They Affect?

In statistics, we have three measures of central tendency namely Mean, Median, and Mode. They help us describe the data.

- Mean is the accurate measure to describe the data when we do not have any outliers present.
- Median is used if there is an outlier in the dataset.
- Mode is used if there is an outlier AND about ½ or more of the data is the same.

Mean' is the only measure of central tendency that is affected by the outliers which in turn impacts Standard deviation.

Example

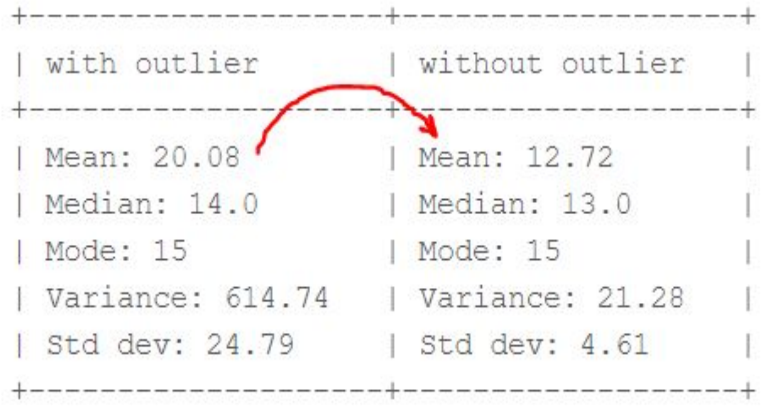
Consider a small dataset, sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9].

By looking at it, one can quickly say ‘101’

is an outlier that is much larger than the other values.

From the calculations, we can clearly say the

Mean is more affected than the Median.



```
+--------------------+-------------------+
| with outlier       | without outlier   |
+--------------------+-------------------+
| Mean: 20.08        | Mean: 12.72       |
| Median: 14.0       | Median: 13.0      |
| Mode: 15           | Mode: 15          |
| Variance: 614.74   | Variance: 21.28   |
| Std dev: 24.79     | Std dev: 4.61     |
+--------------------+-------------------+
```

Detecting Outliers : use visualization and mathematical techniques

Below are some of the techniques of detecting outliers

- Boxplots
- Z-score
- Interquartile Range(IQR)

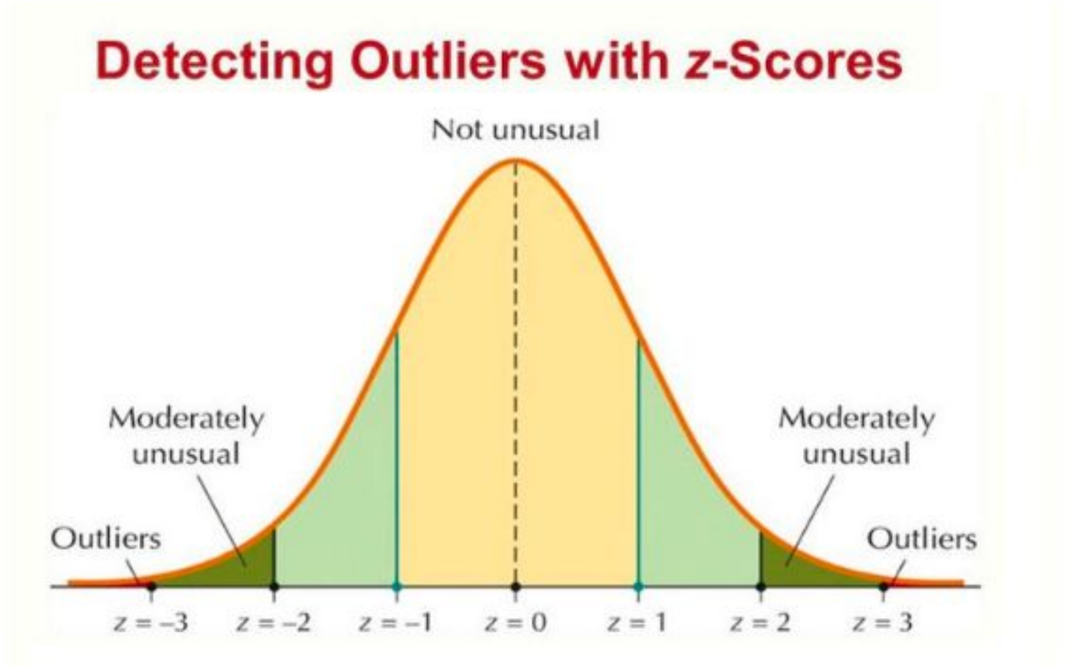
Detecting Outliers Using Boxplot

```
import matplotlib.pyplot as plt


sample= [15, 101, 18, 7, 13, 16, 11, 21, 5, 15, 10, 9]
plt.boxplot(sample, vert=False)
plt.title("Detecting outliers using Boxplot")
plt.xlabel('Sample')
plt.show()
```

Detecting Outliers using the Z-scores

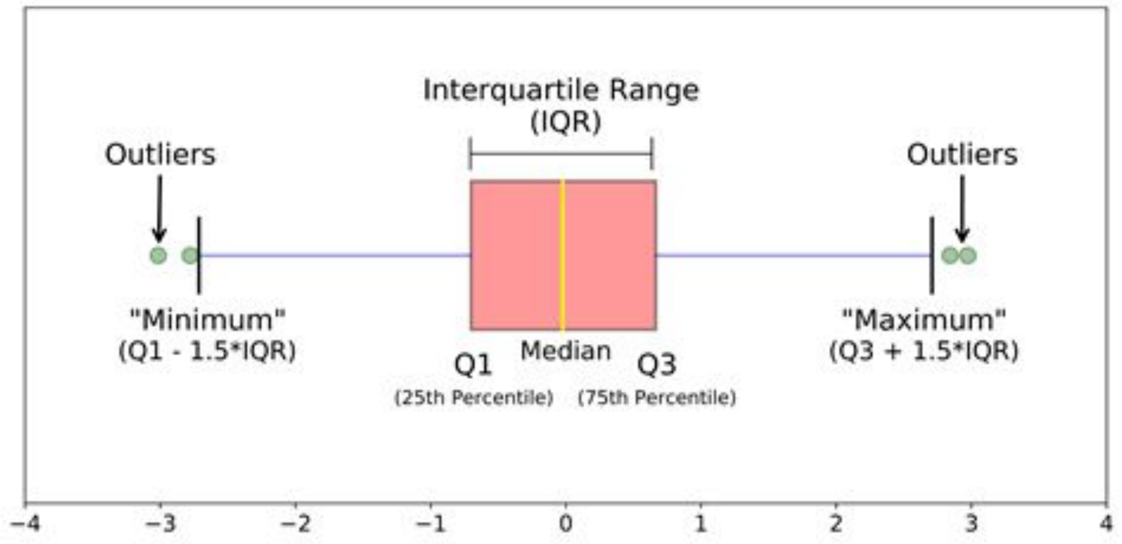
Criteria: any data point whose Z-score falls out of 3rd standard deviation is an outlier.



```
import numpy as np
outliers = []
def detect_outliers_zscore(data):
    thres = 3
    mean = np.mean(data)
    std = np.std(data)
    # print(mean, std)
    for i in data:
        z_score = (i-mean)/std
        if (np.abs(z_score) > thres):
            outliers.append(i)
    return outliers# Driver code
sample_outliers = detect_outliers_zscore(sample)
print("Outliers from Z-scores method: ", sample_outliers)
```

Detecting Outliers using the Interquartile Range(IQR)

Criteria: data points that lie 1.5 times of IQR above Q3 and below Q1 are outliers.



Steps

- Sort the dataset in ascending order
- calculate the 1st and 3rd quartiles(Q1, Q3)

  Q_1 = [(n+1)/4]th term

  Q_2 = [(n+1)/2]th term

  Q_3 = [3(n+1)/4]th term

- compute IQR=Q3-Q1
- compute lower bound = (Q1–1.5*IQR), upper bound = (Q3+1.5*IQR)
- loop through the values of the dataset and check for those who fall below the lower bound and above the upper bound and mark them as outliers

```
outliers = []
def detect_outliers_iqr(data):
    data = sorted(data)
    q1 = np.percentile(data, 25)
    q3 = np.percentile(data, 75)
    # print(q1, q3)
    IQR = q3-q1
    lwr_bound = q1-(1.5*IQR)
    upr_bound = q3+(1.5*IQR)
    # print(lwr_bound, upr_bound)
    for i in data:
        if (i<lwr_bound or i>upr_bound):
            outliers.append(i)
    return outliers# Driver code
sample_outliers = detect_outliers_iqr(sample)
print("Outliers from IQR method: ", sample_outliers)
```

How to Handle Outliers?

1. **Trimming/Remove the outliers**

   we remove the outliers from the dataset. Although it is not a good practice to follow.

   Python code to delete the outlier and copy the rest of the elements to another array.

```
# Trimming
for i in sample_outliers:
  print(i)
  a = np.delete(sample, np.where(sample==i))
  print(a)
```

2. Quantile Based Flooring and Capping

In this technique, the outlier is capped at a certain value above the 90th percentile value or floored at a factor below the 10th percentile value.

```
# Computing 10th, 90th percentiles and replacing the outliers
tenth_percentile = np.percentile(sample,  10)
ninetieth_percentile = np.percentile(sample,  90)
# print(tenth_percentile, ninetieth_percentile)
b = np.where(sample<tenth_percentile, tenth_percentile, sample)
b = np.where(b>ninetieth_percentile, ninetieth_percentile, b)
# print("Sample:", sample)
print("New array:",b)
```

The data points that are lesser than the 10th percentile are replaced with the 10th percentile value and the data points that are greater than the 90th percentile are replaced with 90th percentile value.

3. Mean/Median Imputation

As the mean value is highly influenced by the outliers, it is advised to replace the outliers with the median value.

```
median = np.median(sample)
print(median)
# Replace with median
for i in sample_outliers:
    c = np.where(sample==i, 14, sample)
    print("Sample: ", sample)
    print("New array: ",c)
        # print(x.dtype)
```

Identifying and addressing outliers is paramount in data analysis. These data anomalies can skew results, leading to inaccurate insights and decisions.

Outliers, once understood and managed, become valuable sources of information, ultimately contributing to more informed and reliable decision-making processes.

Data Validation

Data validation is the process of checking, cleaning, and ensuring the accuracy, consistency, and relevance of data before it is used for analysis, reporting, or decision-making.

This process is essential for maintaining data integrity, as it helps identify and correct errors, inconsistencies, and inaccuracies in the data

Types of Data Validation

- **Data type check**: This check verifies that the data entered has the correct data type, such as numerical, text, date, etc. For example, a field that only accepts numerical data should reject any data containing letters or symbols.

- **Code check**: This check ensures that a field is selected from a valid list of values or follows certain formatting rules. For example, a postal code should match a list of valid codes or follow a specific pattern, such as five digits or two letters followed by three digits.

- **Range check**: This check verifies whether the input data falls within a predefined range. For example, a latitude value should be between -90 and 90, while a longitude value should be between -180 and 180. Any values outside this range are invalid.

- **Format check**: This check confirms that the data follows a certain predefined format. For example, a date column should be stored in a consistent format, such as YYYY-MM-DD or DD-MM-YYYY. This helps maintain consistency across data and time.

- **Consistency check**: This check is a type of logical check that confirms that the data is entered in a logically consistent way. For example, a delivery date should be after the shipping date for a parcel, or a customer's age should not be negative.

- **Uniqueness check**: This check ensures that some data, such as IDs or email addresses, are unique by nature and do not have duplicate entries in the database.

Techniques and Tools for Data Validation

- **Manual validation**: This technique involves human inspection and verification of the data. This can be done by using spreadsheets, databases, or other software applications that allow data entry and manipulation. Manual validation is suitable for small and simple data sets, but it can be time-consuming, error-prone, and inefficient for large and complex data sets.

- **Automated validation**: This technique involves using code or specific data validation tools to perform data validation. This can be done by using programming languages, such as Python or R, or data validation packages, such as Google Data Validation Tool, DataTest, Colander, or Voluptuous. Automated validation is suitable for large and complex data sets, as it can save time, reduce errors, and improve efficiency. However, it requires technical skills and knowledge to write and execute the code or use the tools.

Benefits of Data Validation for Data Analytics

- **Improving data quality**: Data validation can help improve the quality of the data by removing or correcting errors, inconsistencies, and inaccuracies. This can enhance the reliability, validity, and usability of the data for analysis, reporting, or decision-making.

- **Reducing data-related risks**: Data validation can help reduce the risks of false or misleading results, faulty decisions. This can prevent potential losses, damages, or reputational harm for the data users or stakeholders.

- **Increasing data efficiency**: Data validation can help increase the efficiency of the data by ensuring that the data is clean, consistent, and relevant. This can reduce the time and effort required for data preparation, transformation, and storage, and enable faster and smoother data analysis, reporting, or decision-making.

Data Transformation

Data transformation is the process of converting, cleansing, and structuring data into a usable format that can be analyzed to support decision making processes, and to propel the growth of an organization.

Data transformation is used when data needs to be converted to match that of the destination system.

Why we need?

we have datasets in which different columns have different units – like one column can be in kilograms, while another column can be in centimeters. Furthermore, we can have columns like income which can range from 20,000 to 100,000, and even more; while an age column which can range from 0 to 100(at the most). Thus, Income is about 1,000 times larger than age.

When we feed these features to the model as is, there is every chance that the income will influence the result more due to its larger value. But this doesn't necessarily mean it is more important as a predictor. So, to give importance to both Age, and Income, we need feature scaling/transformation.

1. Scaling- MinMax Scalar, Standard Scalar
2. Log Transform

Note: Refer colab notebook - Unit 2 Data Transformation(Scaling n Log transformation).ipynb

Data reduction

Data reduction is the process in which an organization sets out to limit the amount of data it’s storing.

Data reduction techniques seek to lessen the redundancy found in the original data set so that large amounts of originally sourced data can be more efficiently stored as reduced data.

Benefits of data reduction

When an organization reduces the volume of data it’s carrying, that company typically realizes substantial financial savings in the form of reduced storage costs associated with consuming less storage space.

Types of data reduction

Dimensionality reduction

Dimensionality refers to the number of attributes (or features) assigned to a single dataset.

The greater the amount of dimensionality, the more data storage demanded by that dataset.

Furthermore, the higher the dimensionality, the more often data tends to be sparse, complicating necessary outlier analysis.

A prime example of dimensionality reduction is the wavelet transform method, which assists image compression by maintaining the relative distance that exists between objects at various resolution levels.

Feature extraction is another possible transformation for data—one that changes original data into numeric features and works in conjunction with machine learning. It differs from principal component analysis (PCA), another means of reducing the dimensionality of large data sets, in which a sizable set of variables is transformed into a smaller set while retaining most of the data from the large set.

Data compression

In order to limit file size and achieve successful data compression, various types of encoding can be used. In general, data compression techniques are considered as either using lossless compression or lossy compression, and they are grouped according to those two types. In lossless compression, data size is reduced through encoding techniques and algorithms, and the complete original data can be restored if needed. Lossy compression, on the other hand, uses other methods to perform its compression, and although its processed data may be worth retaining, it will not be an exact copy, as you would get with lossless compression.

Data preprocessing

Some data needs to be cleaned, treated and processed before it undergoes the data analysis and data reduction processes. Part of that transformation may involve changing the data from analog in nature to digital. Binning is another example of data preprocessing, one in which median values are utilized to normalize various types of data and ensure data integrity across the board.

Normalization

The goal of normalization is to transform features to be on a similar scale. This improves the performance and training stability of the model.

Four common normalization techniques may be useful:

- scaling to a range
- clipping
- log scaling
- z-score

1. Scaling to a range

scaling means converting floating-point feature values from their natural range (for example, 100 to 900) into a standard range—usually 0 and 1 (or sometimes -1 to +1).

Scaling to a range is a good choice when both of the following conditions are met:

- You know the approximate upper and lower bounds on your data with few or no outliers.
- Your data is approximately uniformly distributed across that range.

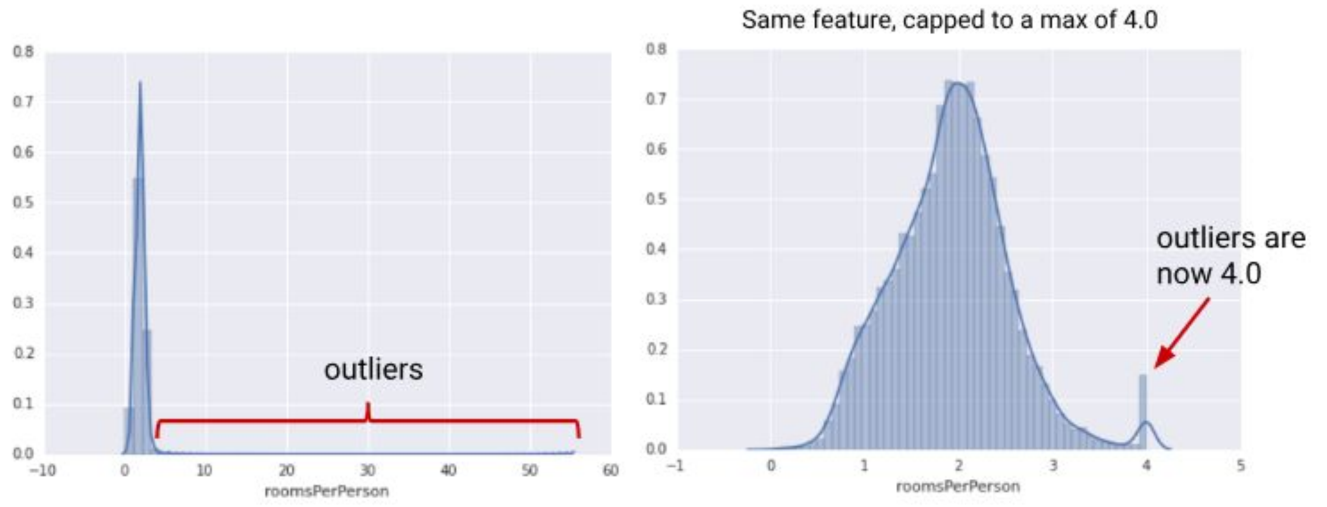
A good example is age. Most age values falls between 0 and 90, and every part of the range has a substantial number of people.

In contrast, you would *not* use scaling on income, because only a few people have very high incomes. The upper bound of the linear scale for income would be very high, and most people would be squeezed into a small part of the scale.

2. Feature Clipping

If your data set contains extreme outliers, you might try feature clipping, which caps all feature values above (or below) a certain value to fixed value. For example, you could clip all temperature values above 40 to be exactly 40.

You may apply feature clipping before or after other normalizations. Another simple clipping strategy is to clip by z-score



3. Log Scaling

Log scaling computes the log of your values to compress a wide range to a narrow range.

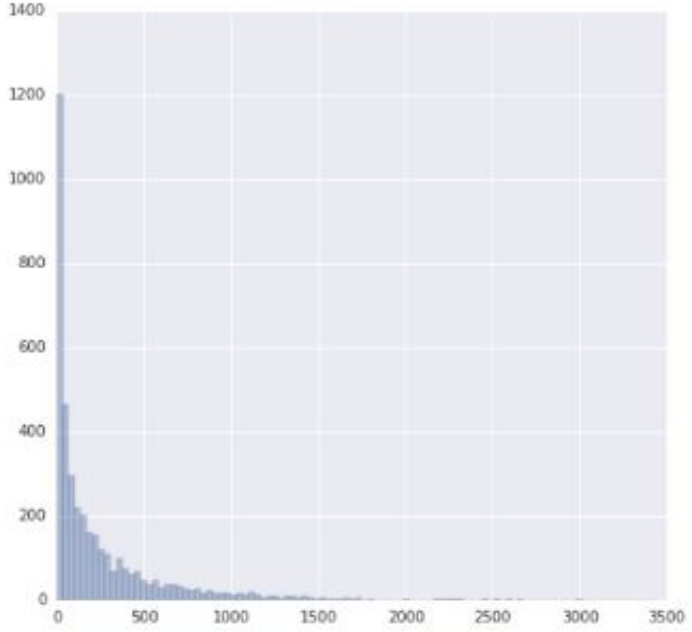
$$x' = log(x)$$

Log scaling is helpful when a handful of your values have many points, while most other values have few points.

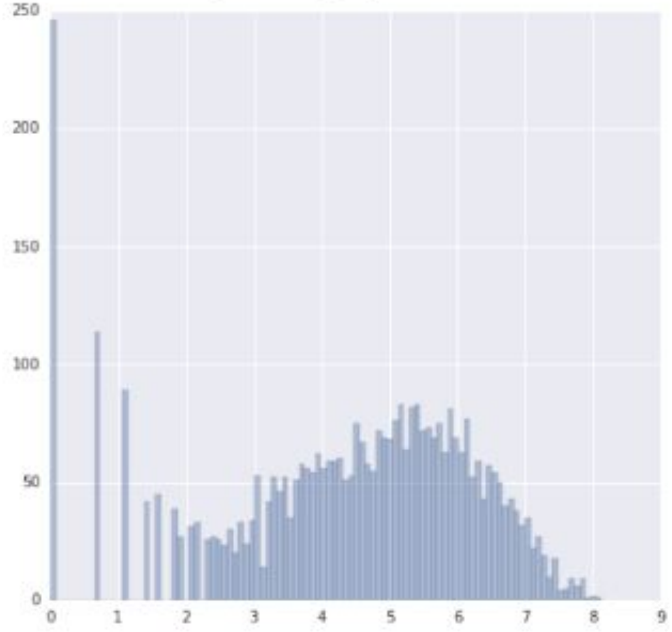
Movie ratings are a good example. In the chart below, most movies have very few ratings (the data in the tail), while a few have lots of ratings (the data in the head).

Log scaling changes the distribution, helping to improve linear model performance.

Ratings per movie



Log ratings per movie

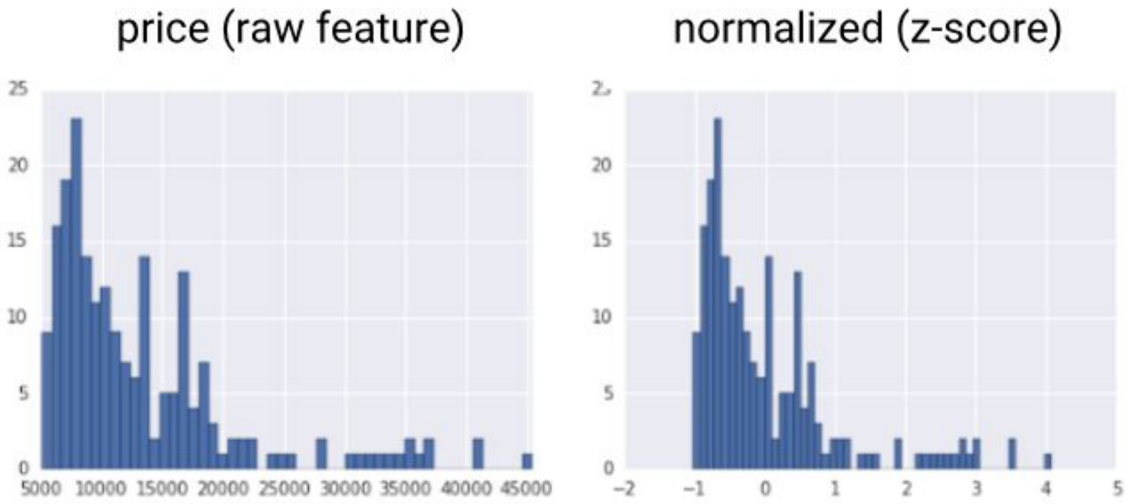


4. **Z-Score**

Z-score is a variation of scaling that represents the number of standard deviations away from the mean. You would use z-score to ensure your feature distributions have mean = 0 and std = 1. It’s useful when there are a few outliers, but not so extreme that you need clipping.

The formula for calculating the z-score of a point, *x*, is as follows:

$$x' = (x - \mu)/\sigma$$



Normalization Technique	Formula	When to Use
Linear Scaling	$x' = (x - x_{min})/(x_{max} - x_{min})$	When the feature is more-or-less uniformly distributed across a fixed range.
Clipping	if $x > max$, then $x' = max$. if $x < min$, then $x' = min$	When the feature contains some extreme outliers.
Log Scaling	$x' = \log(x)$	When the feature conforms to the power law.
Z-score	$x' = (x - \mu) / \sigma$	When the feature distribution does not contain extreme outliers.