PREDICTING HOUSE PRICE USING MACHINE LEARNING

House Price Prediction using Machine Learning

So to deal with this kind of issues Today we will be preparing a MACHINE LEARNING Based model, trained on the House Price Prediction Dataset.

The dataset contains **13 features** :

1	Id	To count the records.
2	MSSubClass	Identifies the type of dwelling involved in the sale.
3	MSZoning	Identifies the general zoning classification of the sale.
4	LotArea	Lot size in square feet.
5	LotConfig	Configuration of the lot
6	BldgType	Type of dwelling
7	OverallCond	Rates the overall condition of the house

8	YearBuilt	Original construction year
9	YearRemodAdd	Remodel date (same as construction date if no remodeling or additions).
10	Exterior1st	Exterior covering on house
11	BsmtFinSF2	Type 2 finished square feet.
12	TotalBsmtSF	Total square feet of basement area
13	SalePrice	To be predicted

Importing Libraries and Dataset

Here we are using

- **Pandas** – To load the Dataframe
- **Matplotlib** – To visualize the data features i.e. barplot
- **Seaborn** – To see the correlation between features using heatmap

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

dataset = pd.read_excel("HousePricePrediction.xlsx")

# Printing first 5 records of the dataset
print(dataset.head(5))
```

Output:

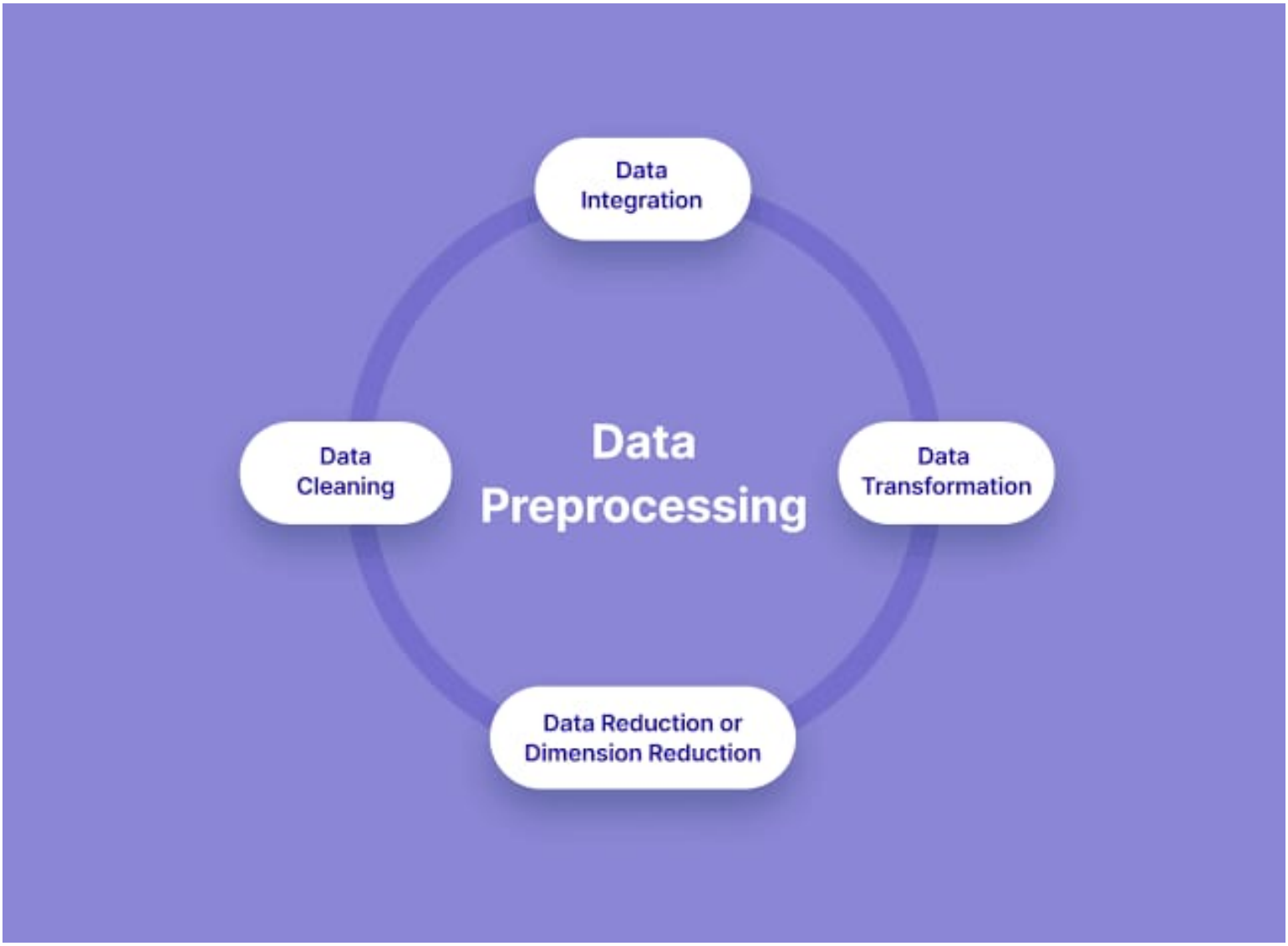
	MSSubClass	MSZoning	LotArea	LotConfig	BldgType	OverallCond	YearBuilt
0	60	RL	8450	Inside	1Fam	5	2003
1	20	RL	9600	FR2	1Fam	8	1976
2	60	RL	11250	Inside	1Fam	5	2001
3	70	RL	9550	Corner	1Fam	5	1915
4	60	RL	14260	FR2	1Fam	5	2000

	YearRemodAdd	Exterior1st	BsmtFinSF2	TotalBsmtSF	SalePrice
0	2003	VinylSd	0.0	856.0	208500.0
1	1976	MetalSd	0.0	1262.0	181500.0
2	2002	VinylSd	0.0	920.0	223500.0
3	1970	Wd Sdng	0.0	756.0	140000.0
4	2000	VinylSd	0.0	1145.0	250000.0

Data Preprocessing:

Data preprocessing is a predominant step in machine learning to yield highly accurate and insightful results. Greater the quality of data, greater is the reliance on the produced results. **Incomplete, noisy, and inconsistent data** are the properties of large real-world datasets. Data preprocessing helps in increasing the quality of data by filling in missing incomplete data, smoothing noise and resolving inconsistencies.

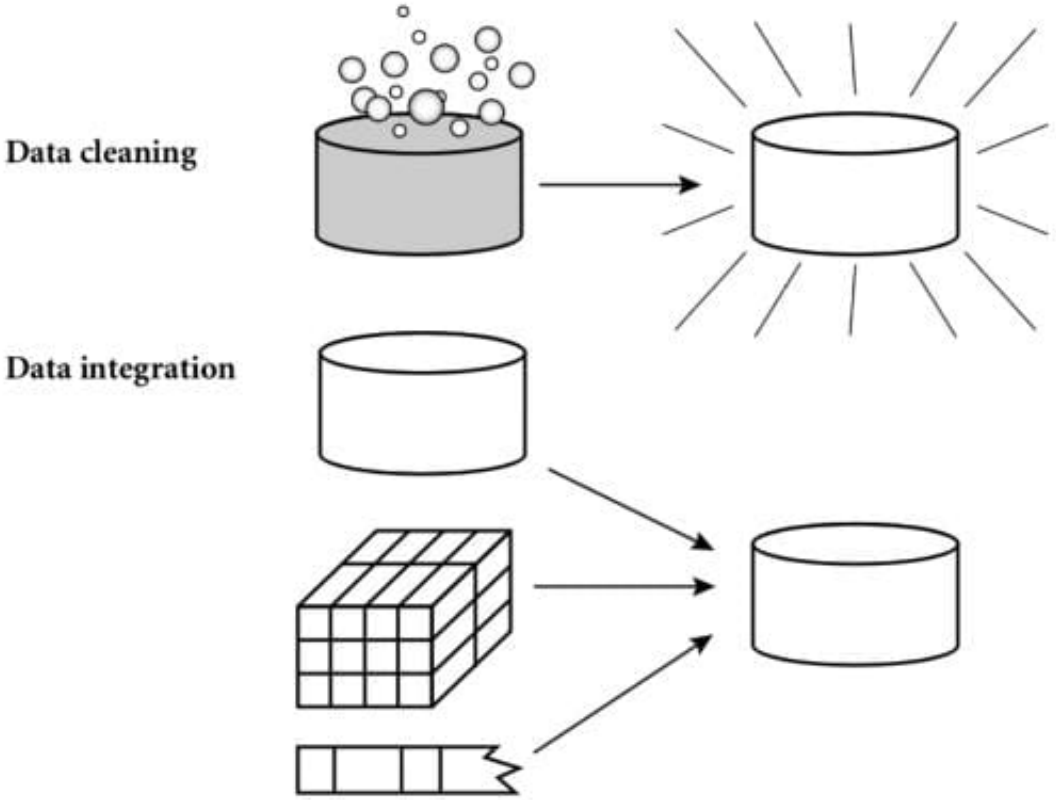
- **Incomplete data** can occur for a number of reasons. Attributes of interest may not always be available, such as customer information for sales transaction data. Relevant data may not be recorded due to a misunderstanding, or because of equipment malfunctions.


Data Integration
Data Transformation
Data Reduction or Dimension Reduction
Data Cleaning
Data Preprocessing

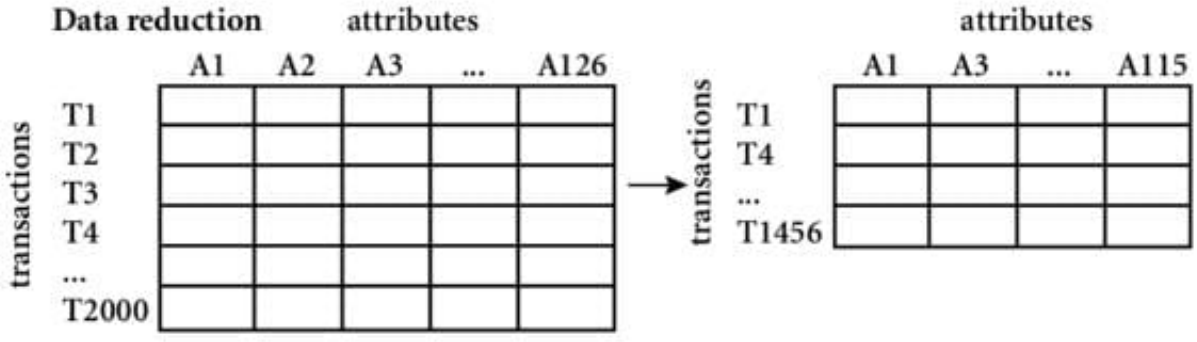
- There are many possible reasons for **noisy data** (having incorrect attribute values). The data collection instruments used may be faulty. There may have been human or computer errors occurring at data entry. Errors in data transmission can also occur. Incorrect data may also result from inconsistencies in naming conventions or data codes used, or inconsistent formats for input fields, such as date.

There are a number of data preprocessing techniques available such as,

1. **Data Cleaning**
2. **Data Integration**
3. **Data Transformation**
4. **Data Reduction**



Data transformation $-2, 32, 100, 59, 48 \longrightarrow -0.02, 0.32, 1.00, 0.59, 0.48$



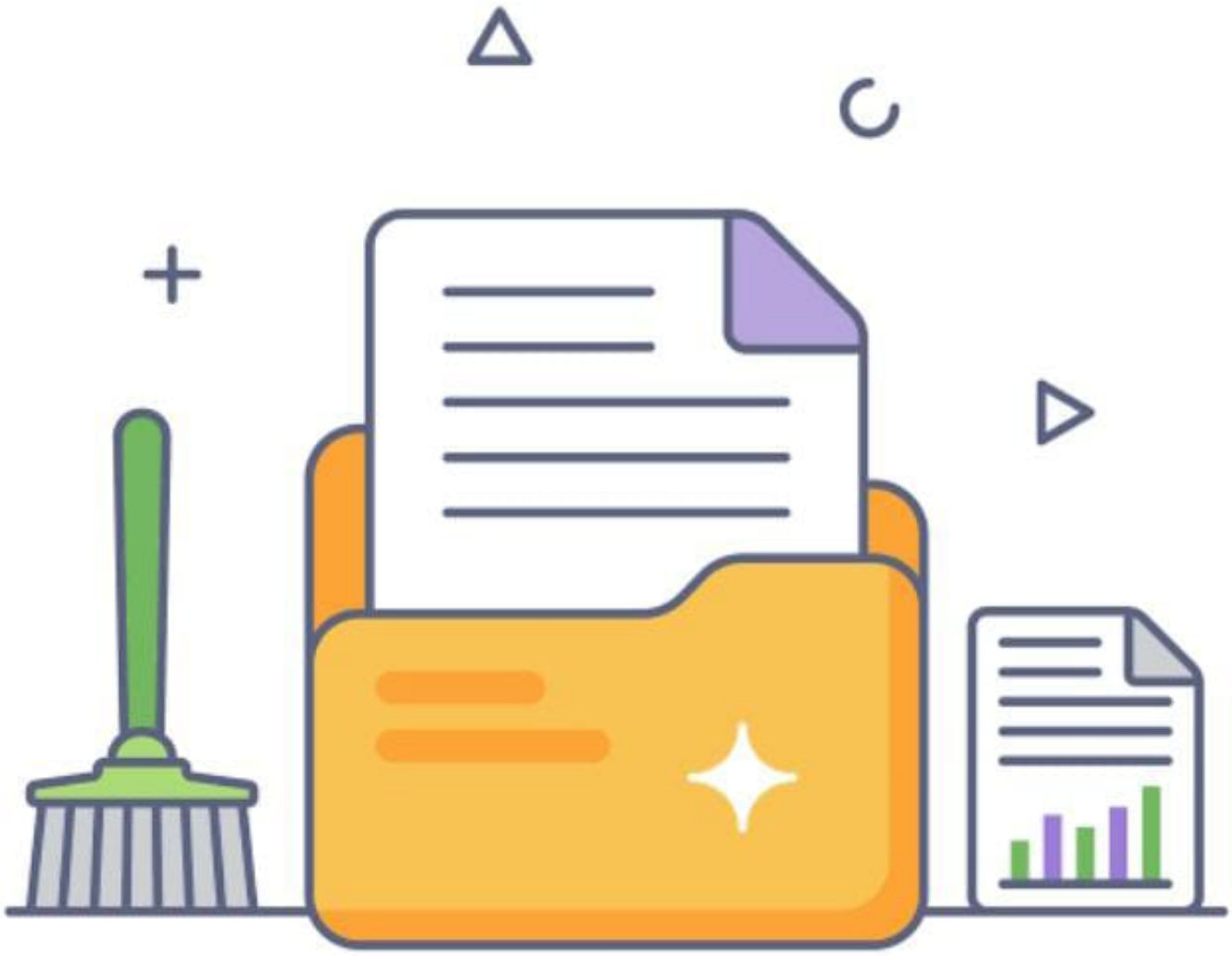
- **Data cleaning** can be applied to filling in missing values, remove noise, resolving inconsistencies, identifying and removing outliers in the data.
- **Data integration** merges data from multiple sources into a coherent data store, such as a data warehouse.
- **Data transformations**, such as normalization, may be applied. For example, normalization may improve the accuracy and efficiency of mining algorithms involving distance measurements.
- **Data reduction** can reduce the data size by eliminating redundant features, or clustering, for instance.

```
int_ = (dataset.dtypes == 'int')
num_cols = list(int_[int_].index)
print("Integer variables:",len(num_cols))

fl = (dataset.dtypes == 'float')
fl_cols = list(fl[fl].index)
print("Float variables:",len(fl_cols))
```

Output:

```
Categorical variables : 4
Integer variables : 6
Float variables : 3
```



Data Cleaning

Data cleaning is one of the important parts of machine learning. It plays a significant part in building a model. It surely isn't the fanciest part of machine learning and at the same time, there aren't any hidden tricks or secrets to uncover. However, the success or failure of a project relies on proper data cleaning. Professional data scientists usually invest a very large portion of their time in this step because of the belief that **"Better data beats fancier algorithms"**.

If we have a well-cleaned dataset, there are chances that we can get achieve good results with simple algorithms also, which can prove very beneficial at times especially in terms of computation when the dataset size is large. Obviously, different types of data will require different types of cleaning. However, this systematic approach can always serve as a good starting point.

Steps Involved in Data Cleaning

Data cleaning is a crucial step in the machine learning (ML) pipeline, as it involves identifying and removing any missing, duplicate, or irrelevant data. The goal of data cleaning is to ensure that the data is accurate, consistent, and free of errors, as incorrect or inconsistent data can negatively impact the performance of the ML model.

Data cleaning, also known as **data cleansing** or **data preprocessing**, is a crucial step in the data science pipeline that involves identifying and correcting or removing errors, inconsistencies, and inaccuracies in the data to improve its quality and usability. Data cleaning is essential because raw data is often noisy, incomplete, and inconsistent, which can negatively impact the accuracy and reliability of the insights derived from it.

Step 1: Remove duplicate or irrelevant observations

Remove unwanted observations from your dataset, including duplicate observations or irrelevant observations. Duplicate observations will happen most often during data collection. When you combine data sets from multiple places, scrape data, or receive data from clients or multiple departments, there are opportunities to create duplicate data. De-duplication is one of the largest areas to be considered in this process. Irrelevant observations are when you notice observations that do not fit into the specific problem you are trying to analyze. For example, if you want to analyze data regarding millennial customers, but your dataset includes older generations, you might remove those irrelevant observations. This can make analysis more efficient and minimize distraction from your primary target–as well as creating a more manageable and more performant dataset.

Step 2: Fix structural errors

Structural errors are when you measure or transfer data and notice strange naming conventions, typos, or incorrect capitalization. These inconsistencies can cause mislabeled categories or classes. For example, you may find “N/A” and “Not Applicable” both appear, but they should be analyzed as the same category.

Step 3: Filter unwanted outliers

Often, there will be one-off observations where, at a glance, they do not appear to fit within the data you are analyzing. If you have a legitimate reason to remove an outlier, like improper data-entry, doing so will help the performance of the data you are working with. However, sometimes it is the appearance of an outlier that will prove a theory you are working on. Remember: just because an outlier exists, doesn’t mean it is incorrect. This step is needed to determine the validity of that number. If an outlier proves to be irrelevant for analysis or is a mistake, consider removing it.

Step 4: Handle missing data

You can't ignore missing data because many algorithms will not accept missing values. There are a couple of ways to deal with missing data. Neither is optimal, but both can be considered.

1. As a first option, you can drop observations that have missing values, but doing this will drop or lose information, so be mindful of this before you remove it.
2. As a second option, you can input missing values based on other observations; again, there is an opportunity to lose integrity of the data because you may be operating from assumptions and not actual observations.
3. As a third option, you might alter the way the data is used to effectively navigate null values.

Step 5: Validate and QA

At the end of the data cleaning process, you should be able to answer these questions as a part of basic validation:

- Does the data make sense?
- Does the data follow the appropriate rules for its field?
- Does it prove or disprove your working theory, or bring any insight to light?
- Can you find trends in the data to help you form your next theory?
- If not, is that because of a data quality issue?

False conclusions because of incorrect or "dirty" data can inform poor business strategy and decision-making. False conclusions can lead to an embarrassing moment in a reporting meeting when you realize your data doesn't stand up to scrutiny. Before you get there, it is important to create a culture of quality data in your organization. To do this, you should document the tools you might use to create this culture and what data quality means to you.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

df =
pd.read_csv(7"student_data.csv")di
splay(df)
```

	participant_id	name	dob	is_student	target	Q1	Q2	Q3	Q4
0	1de9ea66-70d3-4a1f-8735-df5ef7697fb9	Thomas Crosby	1996-07-14	False	898.10	1.0	5.0	0.0	0.0
1	9da618fd-7bf7-4a4d-9f8f-5ffba5f80a0a	Brandon Reeves	1957-11-28	False	816.18	1.0	8.0	1.0	1.0
2	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	1981-02-01	False	598.24	3.0	8.0	0.0	1.0
3	790b2f7c-b5c3-4ec1-a4ce-01e15560eaba	Carol Jones	1990-09-15	True	860.08	3.0	0.0	1.0	2.0
4	6a8f6926-0a22-4ba8-b442-a1244e2e3761	Robert Thompson	1967-06-07	True	863.46	NaN	0.0	6.0	NaN
...	...	...	...	...	...	...	...	...	...
99	af2fb5fe-b506-4098-bf79-c84e5a2e595e	Barbara Ford	1975-01-21	False	703.04	8.0	4.0	4.0	4.0
100	d30aad4b-4503-4e22-8bc4-621b94398520	Maria Castillo	1981-02-01	False	598.24	3.0	8.0	0.0	1.0
101	6016d60e-b39f-458d-9d1d-fe49869adcef	Edward Hood	1984-01-08	False	500.73	3.0	NaN	2.0	4.0
102	77ff71ba-07d3-470a-a402-ffccf29d469b	Jeffrey Smith	1996-09-27	False	905.71	6.0	2.0	7.0	9.0
103	84754a9e-bd1d-4bec-b25b-69b12a2fbff3	Hayley Hoffman	1958-06-21	False	883.55	3.0	0.0	5.0	2.0

104 rows × 9 columns


DATA INTEGRATION

What is Data Integration?

Data integration is the process of combining data from different sources and making it available in a unified format for analysis, reporting, and decision-making. This process is crucial for organizations that aim to consolidate data from various systems, applications, and databases to gain insights and support data-driven decision-making.

Data integration typically involves the following steps:

1. Data extraction: Retrieving data from different sources, such as databases, applications, APIs, and files.
2. **Data transformation:** Converting and standardizing the extracted data into a common format, resolving discrepancies in data types, formats, and units. This may also involve data cleansing and validation to ensure data quality.
3. **Data loading:** Storing the transformed data into a target system or repository, such as a data warehouse, data lake, or centralized database.

```
# import the pandas library

import pandas as pd

import numpy as np


df = pd.DataFrame(np.random.randn(5, 3), index=['a', 'c', 'e', 'f',

'h'],columns=['one', 'two', 'three'])


df = df.reindex(['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h'])


print df
```

Output:

```
one       two     three

a  0.077988  0.476149  0.965836

b       NaN       NaN       NaN

c -0.390208 -0.551605 -2.301950

d       NaN       NaN       NaN

e -2.000303 -0.788201  1.510072

f -0.930230 -0.670473  1.146615

g       NaN       NaN       NaN

h  0.085100  0.532791  0.887415
```


Data
Transformation

Data transformation is the process of converting raw data into a format or structure that would be more suitable for model building and also data discovery in general. It is an imperative step in feature engineering that facilitates discovering insights. This article will cover techniques of numeric data transformation: log transformation, clipping methods, and data scaling.

Why need data transformation?

- the algorithm is more likely to be biased when the data distribution is skewed
- transforming data into the same scale allows the algorithm to compare the relative relationship between data points better

When to apply data transformation

When implementing supervised algorithms, training data and testing data need to be transformed in the same way. This is usually achieved by feeding the training dataset to building the data transformation algorithm and then apply that algorithm to the test set.

The data are transformed in ways that are ideal for mining the data. The data transformation involves steps that are:

1. Smoothing: It is a process that is used to remove noise from the dataset using some algorithms It allows for highlighting important features present in the dataset. It helps in predicting the patterns. When collecting data, it can be manipulated to eliminate or reduce any variance or any other noise form. The concept behind data smoothing is that it will be able to identify simple changes to help predict different trends and patterns. This serves as a help to analysts or traders who need to look at a lot of data which can often be difficult to digest for finding patterns that they wouldn't see otherwise.

2. Aggregation: Data collection or aggregation is the method of storing and

2. Aggregation: Data collection or aggregation is the method of storing and presenting data in a summary format. The data may be obtained from multiple data sources to integrate these data sources into a data analysis description. This is a crucial step since the accuracy of data analysis insights is highly dependent on the quantity and quality of the data used. Gathering accurate data of high quality and a large enough quantity is necessary to produce relevant results. The collection of data is useful for everything from decisions concerning financing or business strategy of the product, pricing, operations, and marketing strategies. For **example**, Sales, data may be aggregated to compute monthly& annual total amounts.

3. Discretization: It is a process of transforming continuous data into set of small intervals. Most Data Mining activities in the real world require continuous attributes. Yet many of the existing data mining frameworks are unable to handle these attributes. Also, even if a data mining task can manage a continuous attribute, it can significantly improve its efficiency by replacing a constant quality attribute with its discrete values. For **example**, (1-10, 11-20) (age:- young, middle age, senior).

4. Attribute Construction: Where new attributes are created & applied to assist the mining process from the given set of attributes. This simplifies the original data & makes the mining more efficient.

5. Generalization: It converts low-level data attributes to high-level data attributes using concept hierarchy. For Example Age initially in Numerical form (22, 25) is converted into categorical value (young, old). For **example**, Categorical attributes, such as house addresses, may be generalized to higher-level definitions, such as town or country.

6. Normalization: Data normalization involves converting all data variables into a given range. Techniques that are used for normalization are:

- **Min-Max Normalization:**
  - This transforms the original data linearly.
  - Suppose that: min_A is the minima and max_A is the maxima of an attribute, P
  - Where v is the value you want to plot in the new range.
  - v' is the new value you get after normalizing the old value.

- **Z-Score Normalization:**
  - In z-score normalization (or zero-mean normalization) the values of an attribute (A), are normalized based on the mean of A and its standard deviation
  - A value, v, of attribute A is normalized to v' by computing

- **Decimal Scaling:**
  - It normalizes the values of an attribute by changing the position of their decimal points
  - The number of points by which the decimal point is moved can be determined by the absolute maximum value of attribute A.
  - A value, v, of attribute A is normalized to v' by computing
  - where j is the smallest integer such that $Max(|v'|) < 1$.
  - Suppose: Values of an attribute P varies from -99 to 99.
  - The maximum absolute value of P is 99.
  - For normalizing the values we divide the numbers by 100 (i.e., j = 2) or (number of integers in the largest number) so that values come out to be as 0.98, 0.97 and so on.

```
import pandas as pd

import numpy as np

df = pd.DataFrame({'one':[10,20,30,40,50,2000],

'two':[1000,0,30,40,50,60]})

print df.replace({1000:10,2000:60})
```

Output:

one two

0 10 10

1 20 0

2 30 30

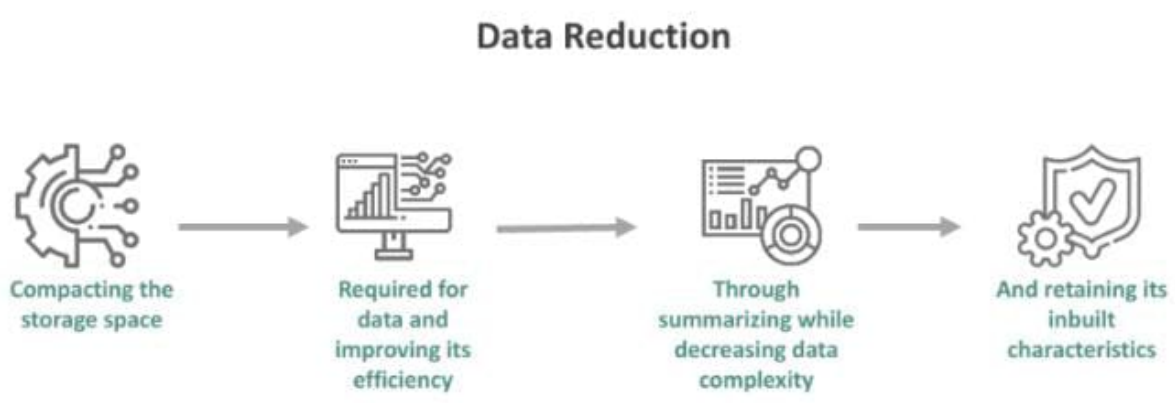
3 40 40

4 50 50

5 60 60

What Is Data Reduction?

> *Data reduction refers to compacting the storage space required for data and improving its efficiency through summarizing while decreasing data complexity and retaining its inbuilt characteristics. Its main purpose is the optimization of storage space, enhancement of data processing speed, and improvement of data for computational efficiency.*



Dimensionality Reduction Algorithms

There are many algorithms that can be used for dimensionality reduction.

Two main classes of methods are those drawn from linear algebra and those drawn from manifold learning.

Linear Algebra Methods

Matrix factorization methods drawn from the field of linear algebra can be used for dimensionality.

Some of the more popular methods include:

- Principal Components Analysis
- Singular Value Decomposition
- Non-Negative Matrix Factorization

Manifold Learning Methods

Manifold learning methods seek a lower-dimensional projection of high dimensional input that captures the salient properties of the input data.

Some of the more popular methods include:

- Isomap Embedding
- Locally Linear Embedding
- Multidimensional Scaling
- Spectral Embedding
- t-distributed Stochastic Neighbor Embedding

Each algorithm offers a different approach to the challenge of discovering natural relationships in data at lower dimensions.

There is no best dimensionality reduction algorithm, and no easy way to find the best algorithm for your data without using controlled experiments.

a

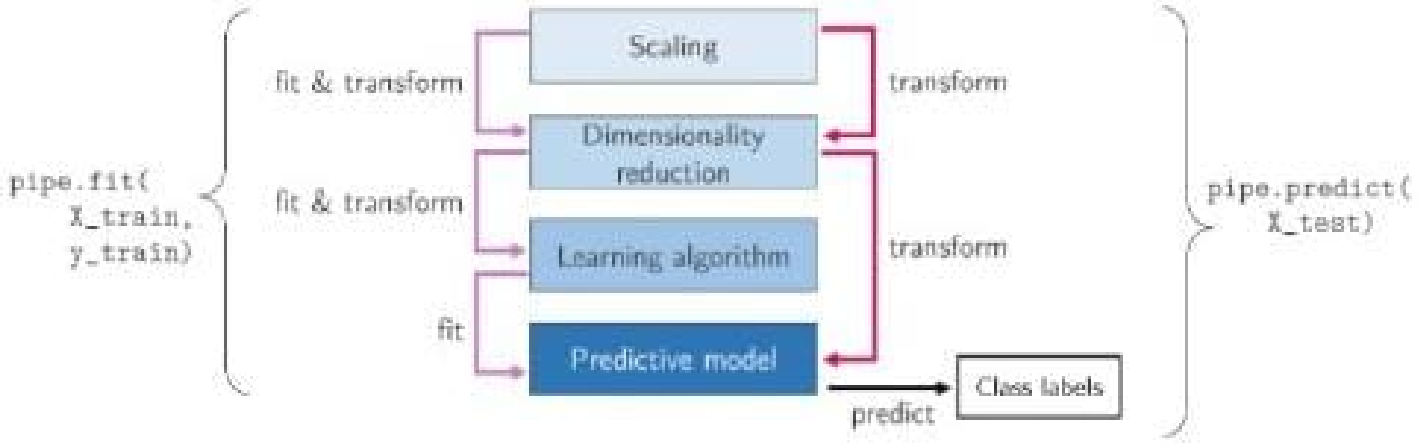
```
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.svm import SVC
from sklearn.pipeline import make_pipeline
from sklearn import datasets
from sklearn.model_selection import train_test_split

iris = datasets.load_iris()
X, y = iris.data, iris.target
X_train, X_test, y_train, y_test =\
    train_test_split(X, y, test_size=0.3,
                     random_state=42, stratify=y)

pipe = make_pipeline(StandardScaler(),
                     PCA(n_components=2),
                     SVC(kernel='linear'))

pipe.fit(X_train, y_train)
y_pred = pipe.predict(X_test)
print('Test Accuracy: %.3f' % pipe.score(X_test, y_test))
```

b



CONCLUSION

Thus the machine learning model to predict the house price based ongiven dataset is executed successfully using xg regressor (a upgraded/slighted boosted form of regular linear regression, this gives lesser error). This model further helps people understand whether this place is more suited for them based on heatmap correlation. It also helps people looking to sell a house at best time for greater profit. Any house price in any location can be predicted with minimum errorby giving appropriate dataset.