

Data Preprocessing

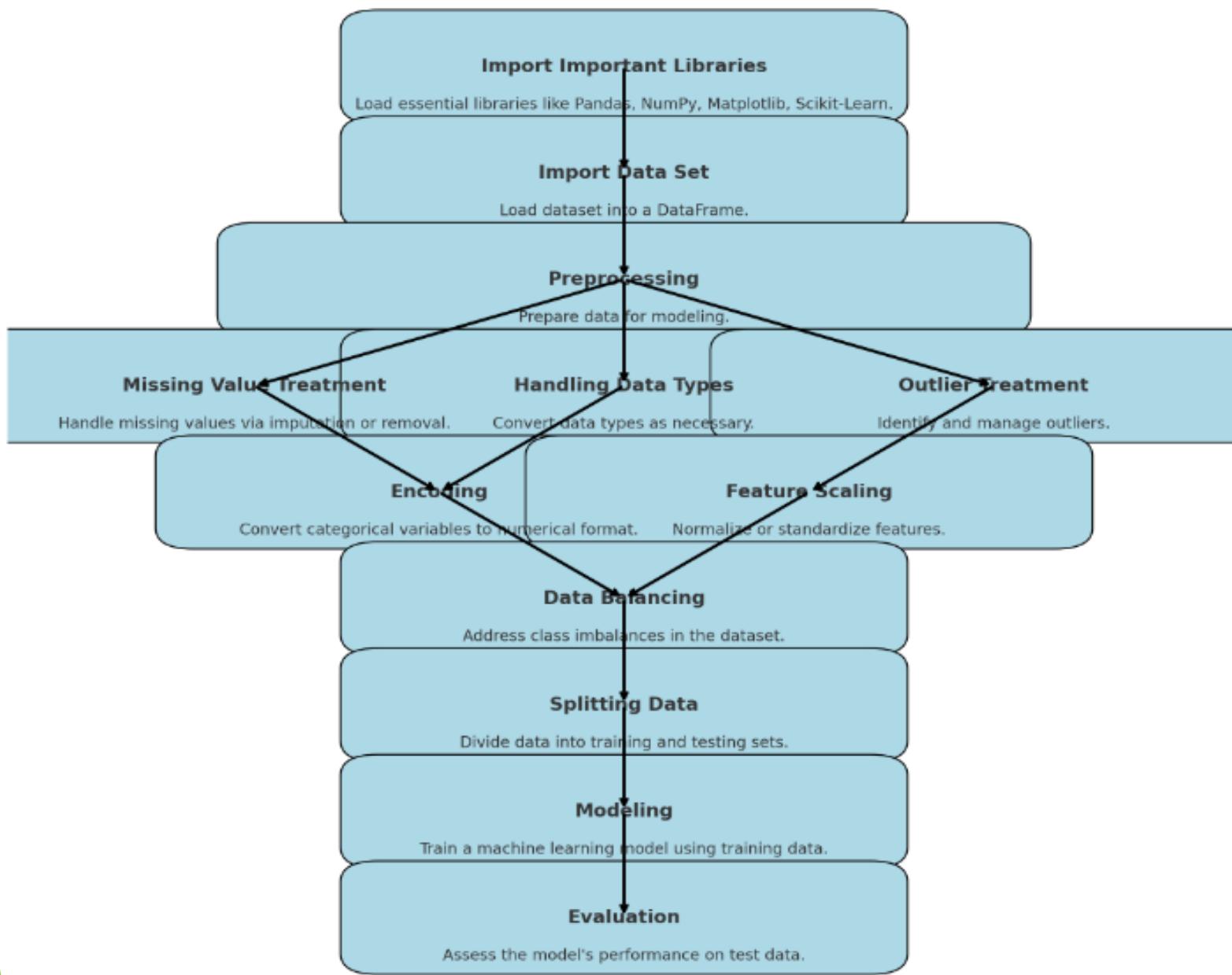
Essential Steps for Preparing Data Before Modeling

Introduction

- ▶ **Data preprocessing** is a crucial step in the machine learning and statistical analysis pipeline.
- ▶ It involves transforming raw data into a clean and usable format, ensuring that the data is consistent, accurate, and relevant for the analysis.
- ▶ Here are the key reasons why data preprocessing is essential:
 - ▶ Improving Data Quality
 - ▶ Enhancing Model Performance
 - ▶ Improving Interpretability
 - ▶ Ensuring Consistency

Basic Steps in Data Preprocessing

- ▶ Step 1 :Import important libraries
- ▶ Step 2: Import dataset
- ▶ Step 3: Preprocessing:
 - ▶ Find duplicates
 - ▶ Missing value treatment
 - ▶ Encoding
 - ▶ Handling data types
 - ▶ Outlier treatment
 - ▶ Feature scaling
 - ▶ Data balancing



Import Important Libraries

```
import os, sys
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
sns.set()

import warnings
warnings.filterwarnings('ignore')

pd.set_option('display.max_rows',500)
pd.set_option('display.max_columns',50)
pd.set_option('display.width',1000)
```

Purpose of Libraries

- ▶ **os:** Functions to interact with the operating system.
 - ▶ **Example Usage:** `os.listdir()` lists files and directories in the specified path.
- ▶ **numpy:** Support for arrays, matrices, and mathematical functions.
- ▶ **pandas:** Data manipulation and analysis.
- ▶ **matplotlib & seaborn:** Data visualization.
- ▶ **warnings:** Manage warning messages in code.
- ▶ **sns.set():** Automatically sets the seaborn plot aesthetics to a default theme.
- ▶ **%matplotlib inline:** A magic command used in Jupyter notebooks to display matplotlib plots inline within the notebook.

Importing Data

- ▶ `Data = pd.read_csv(r"C:\Desktop\DataScience\data.csv")`
- ▶ `Data = pd.read_csv(r"C://Desktop//DataScience//data.csv")`
- ▶ `Data.head()`
- ▶ `Data.tail()`

Finding and Handling Duplicate

- If there is any kind of repetition of data in dataset than it is required to remove them for healthy analysis and prediction.

```
[86]: titanic = pd.read_csv('titanic_dataset.csv')
       titanic.duplicated().sum()
```

```
[39]: titanic.shape
```

```
[39]: (915, 11)
```

```
[88]: titanic.drop_duplicates(keep='first',inplace=True)
```

```
[90]: titanic.shape
```

```
[90]: (891, 11)
```

Handling Duplicates

```
[125]: df.duplicated().sum()
```

```
[125]: 483
```

```
[128]: df[df.duplicated()]
```

```
[131]: df.drop_duplicates(ignore_index=True,inplace=True)
```

```
[132]: df.shape
```

```
[132]: (10357, 13)
```

```
[134]: df.duplicated().sum()
```

```
[134]: 0
```

Handling Missing Values

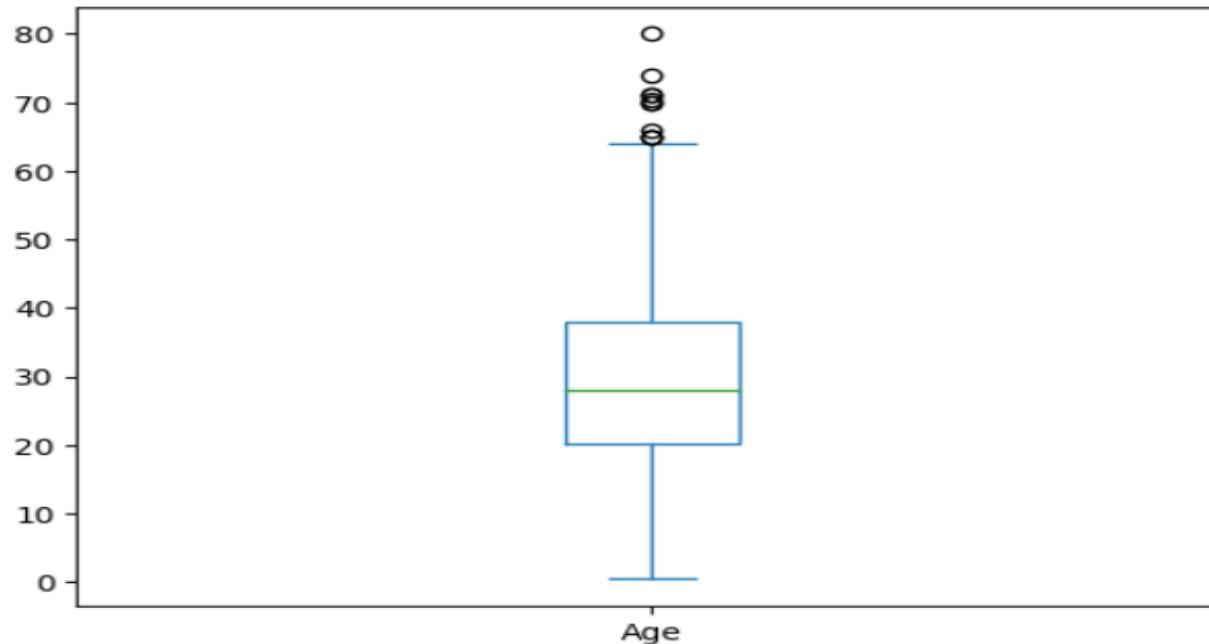
- ▶ Identifying missing values using `df.isnull().sum()`.
- ▶ In percentage form: `df.isnull().sum()/len((df)*100`
- ▶ Techniques to handle missing values:
- ▶ If any variable have missing value > 25% , drop it.
 - ▶ `data = data.drop(['name of column'], axis = 1)`
- ▶ - Else Imputing missing values

Imputation Method

- ▶ Various imputation Approaches are:
- ▶ Simple Statistical Imputation:
 - ▶ Mean: If no outliers.
 - ▶ Median:If data have outliers.
 - ▶ Mode : If variables are categorical type.
- ▶ KNN imputation: Replaces missing values based on the mean or median of the nearest neighbors' values.

```
[17]: dataset['Age'].plot(kind='box')
```

```
[17]: <AxesSubplot:>
```



```
[19]: dataset['Age'] = dataset['Age'].fillna(dataset['Age'].median())
```

[111]:

```
# Embarked : object
titanic['Embarked'].value_counts()
```

[111]:

```
Embarked
S    644
C    168
Q     77
Name: count, dtype: int64
```

[113]:

```
# Filling missing data by "S"
titanic['Embarked'] = titanic['Embarked'].fillna('S')
```

```
[148]: df.Rating.dtype
```

```
[148]: dtype('float64')
```

```
[149]: df.Rating.unique()
```

```
[149]: array([ 4.1,  3.9,  4.7,  4.5,  4.3,  4.4,  3.8,  4.2,  4.6,  3.2,  4. ,  
           nan,  4.8,  4.9,  3.6,  3.7,  3.3,  3.4,  3.5,  3.1,  5. ,  2.6,  
           3. ,  1.9,  2.5,  2.8,  2.7,  1. ,  2.9,  2.3,  2.2,  1.7,  2. ,  
           1.8,  2.4,  1.6,  2.1,  1.4,  1.5,  1.2, 19. ])
```

```
[150]: # - It's a categorical column  
      # - We can fill the null values with the mode value
```

```
[154]: df.Rating.fillna(df.Rating.mode()[0],inplace=True)
```

```
[155]: df.Rating.mode()[0]
```

```
[155]: 4.4
```

Encoding Categorical Variables

- ▶ Encoding categorical variables is a critical step in data preprocessing for machine learning models, as most models require numerical input.
- ▶ There are two approach:
- ▶ **Label encoding and one-hot encoding.**

Label Encoding

- ▶ It converts each category in a categorical variable to a unique integer.
- ▶ **When to Use:** When the categorical variable has an ordinal relationship (e.g., low, medium, high).
- ▶ When there are a limited number of categories.

```
[42]:
```

```
dataset2['Sex'] = dataset2['Sex'].astype('category')
dataset2['Sex'] = dataset2['Sex'].cat.codes
```

```
[46]:
```

```
dataset2.head(10)
```

```
[46]:
```

	Sex	Embarked
0	1	S
1	0	C
2	0	S
3	0	S
4	1	S

[48]:

```
dataset2['Embarked'] = dataset2['Embarked'].astype('category')
dataset2['Embarked'] = dataset2['Embarked'].cat.codes
```

[52]:

```
dataset2.head(10)
```

[52]:

	Sex	Embarked
0	1	2
1	0	0
2	0	2
3	0	2
4	1	2

```
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Sample data
data = {'color': ['red', 'green', 'blue', 'green', 'blue', 'red']}
df = pd.DataFrame(data)

# Initialize and apply LabelEncoder
label_encoder = LabelEncoder()
df['color_encoded'] = label_encoder.fit_transform(df['color'])

print(df)
```

	color	color_encoded
0	red	2
1	green	1
2	blue	0
3	green	1
4	blue	0
5	red	2

One-Hot Encoding

- ▶ One-hot encoding converts categorical variables into a series of binary columns, each representing a unique category.
- ▶ When to Use
 - ▶ When the categorical variable is nominal (no intrinsic order).
 - ▶ When you want to avoid introducing ordinal relationships.

[84]:

```
dataset1 = pd.get_dummies(dataset1, columns=['Sex', 'Embarked'])
```

```
# Dummy variable concept - (n-1)
dataset1 = dataset1.drop(['Sex_male', 'Embarked_C'], axis=1)
```

- ▶ `pd.get_dummies()` is a function in the pandas library in Python used for one-hot encoding categorical variables.
- ▶ In one hot encoding usually drop one column.

[86]:

	Sex_female	Sex_male	Embarked_C	Embarked_Q	Embarked_S
0	False	True	False	False	True
1	True	False	True	False	False
2	True	False	False	False	True
3	True	False	False	False	True
4	False	True	False	False	True

```
from sklearn.preprocessing import OneHotEncoder
import pandas as pd

# Sample data
data = {'color': ['red', 'green', 'blue', 'green', 'blue', 'red']}
df = pd.DataFrame(data)

# Initialize and apply OneHotEncoder
one_hot_encoder = OneHotEncoder(sparse=False)
one_hot_encoded = one_hot_encoder.fit_transform(df[['color']])

# Convert to DataFrame for better readability
df_one_hot = pd.DataFrame(one_hot_encoded, columns=one_hot_encoder.categories_[0])

print(df_one_hot)
```

	blue	green	red
0	0.0	0.0	1.0
1	0.0	1.0	0.0
2	1.0	0.0	0.0
3	0.0	1.0	0.0
4	1.0	0.0	0.0
5	0.0	0.0	1.0

Handling Outlier

- ▶ **Identifying outliers:**
- ▶ **Visualization-Based Detection:**
 - ▶ Box plots
 - ▶ Histograms with normal distribution curve
- **Statistical Methods:**
 - Z-Score (standard deviation approach)
 - IQR (Interquartile Range):
 - Values below $Q1 - 1.5 \times IQR$
 - Values above $Q3 + 1.5 \times IQR$.

Approaches to Handle Outliers

- ▶ Capping method:
 - ▶ Using IQR or Z score
- ▶ Transformation Approach:
 - ▶ Log Transformation
 - ▶ Square root Transformation
 - ▶ Box- Cox Transformation
 - ▶ Winsorization Method

Finding Outliers Using IQR method

```
[23]: # Finding the IQR
```

```
percentile25 = df['placement_exam_marks'].quantile(0.25)
percentile75 = df['placement_exam_marks'].quantile(0.75)
print(percentile25)
print(percentile75)
```

```
17.0
```

```
44.0
```

```
[24]: iqr = percentile75 - percentile25
iqr
```

```
[24]: 27.0
```

```
[25]: upper_limit = percentile75 + 1.5*iqr
lower_limit = percentile25 - 1.5*iqr

print(upper_limit)
print(lower_limit)
```

```
84.5
```

```
-23.5
```

```
[26]: # finding outliers  
df[df['placement_exam_marks'] > upper_limit]
```

```
[26]:
```

	cgpa	placement_exam_marks	placed	cgpa_zscore
9	7.75	94.0	1	1.280667
40	6.60	86.0	1	-0.586526
61	7.51	86.0	0	0.890992
134	6.33	93.0	0	-1.024910
162	7.80	90.0	0	1.361849
283	7.09	87.0	0	0.209061
290	8.38	87.0	0	2.303564

```
[27]: df[df['placement_exam_marks'] < lower_limit]
```

```
[27]:
```

	cgpa	placement_exam_marks	placed	cgpa_zscore
--	------	----------------------	--------	-------------

```
[13]: Q1 = np.percentile(df['cgpa'], 25)
Q3 = np.percentile(df['cgpa'], 75)
IQR = Q3 - Q1
```

```
[220]: pos_outlier = Q3 + 1.5 * IQR
neg_outlier = Q1 - 1.5 * IQR
```

Handling outlier using capping by IQR

```
[30]: new_df_cap = df.copy()

[31]: new_df_cap.shape

[31]: (1000, 4)

[32]: new_df_cap['placement_exam_marks'] = np.where(new_df_cap['placement_exam_marks'] > upper_limit,
                                                 upper_limit,
                                                 np.where(new_df_cap['placement_exam_marks'] < lower_limit,
                                                          lower_limit,
                                                          new_df_cap['placement_exam_marks']))

[33]: new_df_cap.shape

[33]: (1000, 4)

[34]: new_df_cap[new_df_cap['placement_exam_marks'] > upper_limit]

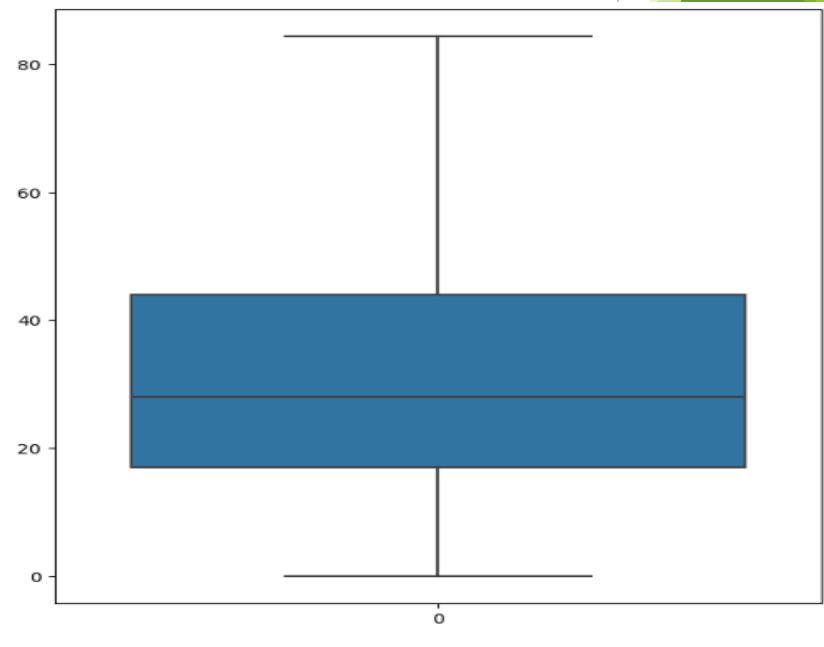
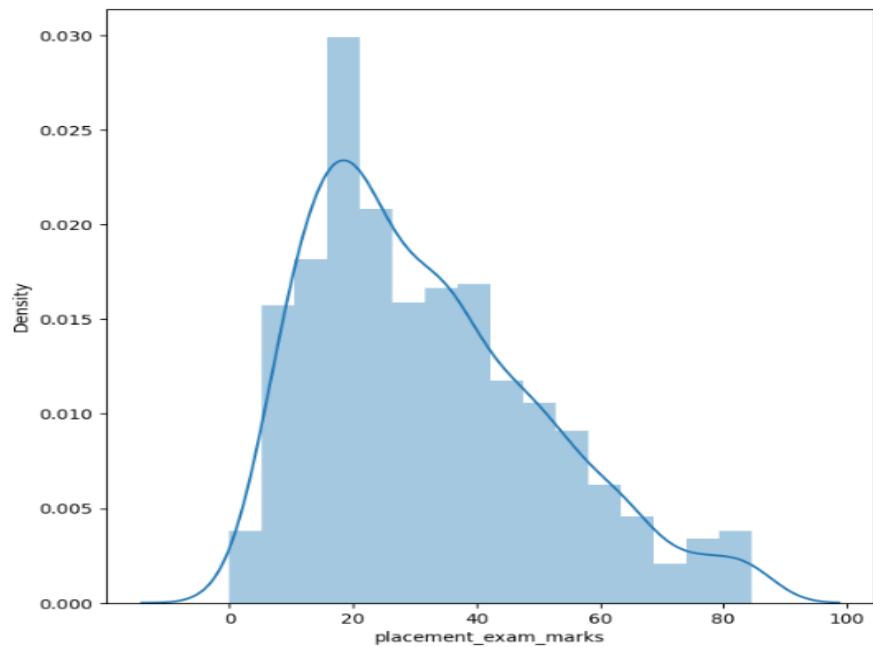
[34]: cgpa placement_exam_marks placed cgpa_zscore
```

Distribution and box plot

```
[36]: plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
sns.distplot(new_df_cap['placement_exam_marks'])

plt.subplot(1,2,2)
sns.boxplot(new_df_cap['placement_exam_marks'])

plt.show()
```



Handling outlier using capping by Z score:

- ▶ Step1: find mean and standard deviation

```
: print("mean value of cgpa", df['cgpa'].mean())
print()
print("std value of cgpa", df['cgpa'].std())
print()
print("min value of cgpa", df['cgpa'].min())
print()
print("max value of cgpa", df['cgpa'].max())

mean value of cgpa 6.961240000000001

std value of cgpa 0.6158978751323894

min value of cgpa 4.89

max value of cgpa 9.12
```

- ▶ Step 2: Find minimum and maximum based on normal distribution parameters to identify an outlier

```
[17]: upper_limit = df['cgpa'].mean() + 3*df['cgpa'].std()  
lower_limit = df['cgpa'].mean() - 3*df['cgpa'].std()
```

```
[18]: print(upper_limit)  
print(lower_limit)
```

```
8.808933625397168  
5.113546374602832
```

Step 3: Capping

```
[19]: df['cgpa'] = np.where(df['cgpa']>upper_limit, upper_limit,  
                           np.where(df['cgpa']<lower_limit, lower_limit, df['cgpa']))  
  
[20]: df['cgpa'].describe()  
  
[20]: count    1000.000000  
mean      6.961499  
std       0.612688  
min      5.113546  
25%      6.550000  
50%      6.960000  
75%      7.370000  
max      8.808934  
Name: cgpa, dtype: float64
```

- To view the outliers based on mini or max

```
df[(df['cgpa'] > 8.80) | (df['cgpa'] < 5.11)]
```

	cgpa	placement_exam_marks	placed
485	4.92	44.0	1
995	8.87	44.0	1
996	9.12	65.0	1
997	4.89	34.0	0
999	4.90	10.0	1

► To Calculate Z-score

```
df['cgpa_zscore'] = (df['cgpa'] - df['cgpa'].mean())/df['cgpa'].std()
```

```
df
```

	cgpa	placement_exam_marks	placed	cgpa_zscore
0	7.19	26.0	1	0.371425
1	7.46	38.0	1	0.809810
2	7.54	40.0	1	0.939701
3	6.42	8.0	1	-0.878782
4	7.23	17.0	0	0.436371

► To view the outliers based on Z-score:

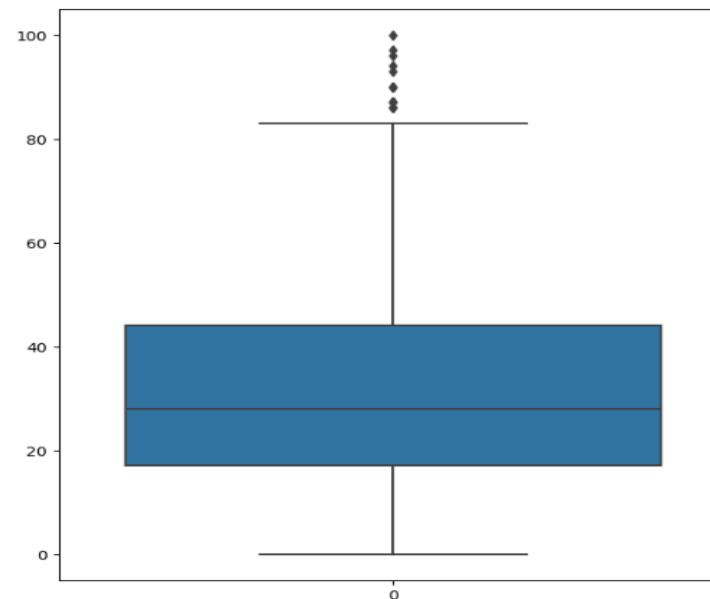
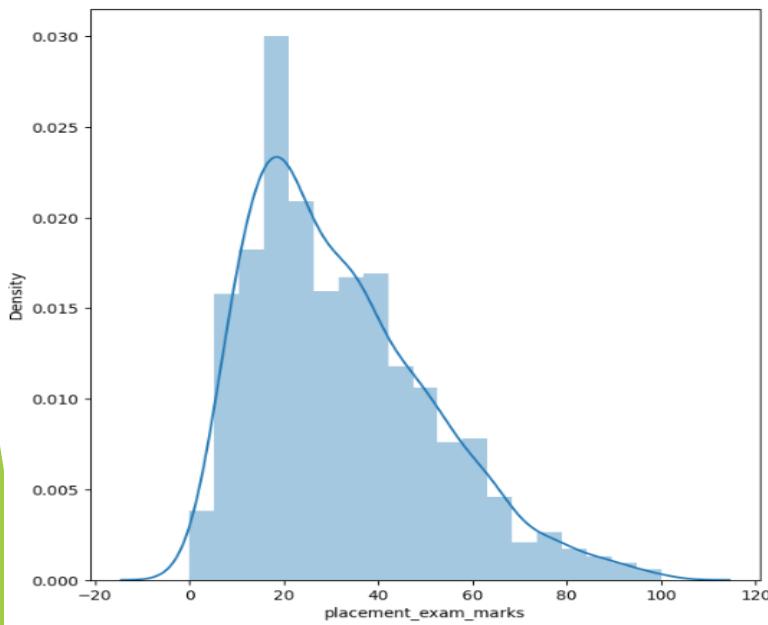
```
df[(df['cgpa_zscore'] > 3) | (df['cgpa_zscore'] < -3)]
```

	cgpa	placement_exam_marks	placed	cgpa_zscore
485	4.92	44.0	1	-3.314251
995	8.87	44.0	1	3.099150
996	9.12	65.0	1	3.505062
997	4.89	34.0	0	-3.362960
999	4.90	10.0	1	-3.346724

To visualize the outliers by distribution plot and boxplot:

```
[35]: # comparing
plt.figure(figsize=(16,8))
plt.subplot(1,2,1)
sns.distplot(df['placement_exam_marks'])

plt.subplot(1,2,2)
sns.boxplot(df['placement_exam_marks'])
```



Log Transformation

- ▶ Logarithmic transformation is often used to reduce the effect of large outliers by compressing the range of data values.
- ▶ **Formula:**
- ▶ $y=\log(x)$
- ▶ Works best when all values are positive (since log of zero or negative values is undefined).
- ▶ Helps stabilize variance and normalize skewed distributions.

```
[10]: df_log_transformation = df.applymap(lambda x: np.log(x) if x > 0 else x)
print(df_log_transformation.describe())
```

	Variable1	Variable2
count	103.000000	103.000000
mean	3.897492	3.450651
std	0.291443	0.262252
min	3.245329	2.988442
25%	3.727106	3.325204
50%	3.887418	3.405378
75%	4.027562	3.523882
max	5.135798	4.787492

Square Root Transform

- ▶ Square root transformation reduces the magnitude of large values but less aggressively than logarithmic transformation.
- ▶ It works with zero values but not negatives.
- ▶ **Formula:**
- ▶ $y=\sqrt{x}$

```
[19]: df_sqrt_transformation = df.applymap(lambda x: np.sqrt(x) if x > 0 else x)
print(df_sqrt_transformation.describe())
```

	Variable1	Variable2
count	103.000000	103.000000
mean	7.100766	5.669671
std	1.196587	0.931304
min	5.066573	4.455864
25%	6.446677	5.273014
50%	6.984609	5.488687
75%	7.491591	5.823731
max	13.038405	10.954451

Box-Cox Transformation

- ▶ The Box-Cox transformation is a family of power transformations that stabilize variance, make the data more normal-like, and reduce the impact of outliers.
- ▶ Unlike logarithmic or square root transformations, the Box-Cox method includes an adjustable parameter (λ) that determines the transformation applied.
- ▶ **When to Use**
- ▶ When data is not normally distributed and transformations like log or square root are insufficient.
- ▶ To stabilize variance across different scales of data.

Mathematical Formula

The Box-Cox transformation is defined as:

$$y(\lambda) = \begin{cases} \frac{y^\lambda - 1}{\lambda}, & \text{if } \lambda \neq 0 \\ \ln(y), & \text{if } \lambda = 0 \end{cases}$$

- λ : Determines the type of transformation. Common values include:
 - $\lambda = 1$: No transformation (identity).
 - $\lambda = 0$: Logarithmic transformation.
 - $\lambda = 0.5$: Square root transformation.
- Box-Cox requires all data values to be positive.

```
df_box_cox_transformation = df.copy()

for col in df.columns:
    df_box_cox_transformation[col], _ = boxcox(df[col] + 1e-6)
    # adding small constant to handle zeros
print(df_box_cox_transformation.describe())
```

	Variable1	Variable2
count	103.000000	103.000000
mean	1.040327	0.595645
std	0.006870	0.000609
min	1.017788	0.593607
25%	1.036425	0.595341
50%	1.040986	0.595629
75%	1.044448	0.595989
max	1.060401	0.597434

Winsorization:

- ▶ Winsorization replaces extreme values with specified percentiles to limit the influence of outliers while retaining the dataset's size.
- ▶ Steps:
- ▶ Define lower and upper limits (e.g., 5th and 95th percentiles).
- ▶ Replace values below the lower limit with the 5th percentile and above the upper limit with the 95th percentile.

```
[59]: df_winsorized = df.apply(lambda x: winsorize(x, limits=[0.05, 0.05]))
print(df_winsorized)
```

	Variable1	Variable2
0	64.458713	28.999760
1	56.530266	24.094480
2	33.324787	31.311350
3	33.324787	29.009641
4	60.190868	37.719077
..
98	57.291343	27.761510
99	58.190403	32.080371
100	69.629120	39.635437
101	69.629120	39.635437
102	69.629120	39.635437

Feature Scaling

- ▶ Importance of scaling features.
- ▶ **Methods:** Standard Scaler, Min-Max Scaler, Normalizer.
- ▶ We do not do feature scaling with dependent variables.
- ▶ So 1st separate the data into independent and dependent variable.

```
x = dataset.iloc[:,1:]
y = dataset[['Survived']]
```

Standardization:

- ▶ Scaling features to have zero mean and unit variance. This is particularly important for algorithms that rely on distance measures, such as SVM and k-NN.

```
from sklearn.preprocessing import StandardScaler  
sc = StandardScaler()  
x = sc.fit_transform(x)  
pd.DataFrame(x)
```

When to Use Standard Scaler

- ▶ The **Standard Scaler** standardizes features by removing the mean and scaling to unit variance.
- ▶ Where "Remove the mean" means that for each feature in your dataset, you subtract the mean (average) value of that feature from all the values of that feature.
- ▶ This process centers the feature around zero, ensuring that the transformed feature has a mean of zero.
- ▶ This means each feature will have a mean of '0' and a standard deviation of 1.
- ▶ This ensures that each feature contributes equally to the model.

- ▶ When your dataset contains features with different units (e.g., age in years, income in dollars, and height in centimeters), StandardScaler helps to bring all features to the same scale.
- ▶ You are using algorithms that assume normal distribution of features.

1. Original feature: `[1, 4, 7]`

2. Mean: $\mu = \frac{1+4+7}{3} = 4$

3. Subtract the mean: `[-3, 0, 3]`

```
X = np.array([[1, 2, 3],  
             [4, 5, 6],  
             [7, 8, 9]])
```

```
X_scaled = [[-1.22474487 -1.22474487 -1.22474487]  
              [ 0. 0. 0.]  
              [ 1.22474487 1.22474487 1.22474487]]
```

When to Use Standard Scaler:

➤ Algorithms that Assume Normal Distribution:

1. **Linear Regression:** Assumes that the relationship between the input and output is linear.
2. **Logistic Regression:** Assumes a linear relationship between the input features and the log-odds of the target.
3. **Linear Discriminant Analysis (LDA):** Assumes data is normally distributed within each class.
4. **Support Vector Machines (SVM):** Assumes features have similar scales for optimal performance.
5. **Principal Component Analysis (PCA):** Assumes the data is centered around the origin for variance maximization.
6. **K-Means Clustering:** Assumes features are on similar scales for effective distance calculation.

Normalizer:

- ▶ Normalizer is suitable when the goal is to scale individual samples to have unit norm(length).
- ▶ This technique is useful when the **direction of the data points** is more important than the magnitude of their distance from the origin.
- ▶ Normalizing the data to unit norm ensures that the focus is on the direction of each sample.
 - ▶ Sample data $X = [[3, 4], [1, 2], [4, 5]]$
 - ▶ **Normalized data: means sum of data point(3,4 = 1).**

$$X = [[0.6 \ 0.8] [0.4472136 \ 0.89442719] [0.62469505 \ 0.78086881]]$$

Each row vector in `X_normalized` has a length of 1. For example:

$$\sqrt{0.6^2 + 0.8^2} = \sqrt{0.36 + 0.64} = \sqrt{1} = 1$$

When to Use Normalizer:

1. Feature Comparison:

- **Cosine Similarity:** When you want to measure the cosine similarity between samples. By normalizing, you ensure that the angle between vectors becomes the metric rather than their magnitude.
- **Clustering:** When using clustering algorithms like K-Means, normalized data can improve the convergence speed and cluster quality, especially if the data has different scales.
- **Nearest Neighbor:** When you want to perform k-nearest neighbors (KNN) classification or regression, normalizing ensures that all features contribute equally to the distance calculations.

4 Sparse Data:

- When working with sparse data (data with a lot of zeros), normalizing can make algorithms like Support Vector Machines (SVM) and Principal Component Analysis (PCA) perform better by ensuring that features with more non-zero values don't dominate.

5 Text Data:

- **TF-IDF:** In Natural Language Processing (NLP), TF-IDF vectors are often normalized to have unit norm to account for the difference in document lengths and to focus on the relative importance of terms.

```
from sklearn.preprocessing import Normalizer  
nor = Normalizer()  
x1 = nor.fit_transform(x1)  
pd.DataFrame(x1)
```

Concept of `fit_transform` and `transform`:

- ▶ **`fit_transform`**: Use this on your training data to compute the necessary parameters(mean, standard deviation) and apply the transformation in one step.
 - ▶ **`fit`** the preprocessing transformers on the training data to learn the necessary parameters.
 - ▶ **`transform`** the training data using the fitted transformers.
-
- ▶ **`transform`**: Use this on your test data (or any new data) to apply the transformation using the parameters computed from the training data.
 - ▶ For test data we do not use ‘`fit`’ and we are using the same parameters as calculated for training.

- ▶ Converting data types using `'pd.to_numeric'`.

Common Data Types and Their Handling

- ▶ Numerical Data:
 - ▶ Integers: Whole numbers.
 - ▶ Floats: Decimal numbers.
 - ▶ Handling: Ensure numerical columns are in the correct format and handle missing values appropriately.

Numerical Data:

```
[11]: import pandas as pd

df = pd.DataFrame({'integers': [1, 2, 3], 'floats': [1.1, 2.2, 3.3]})

df['integers'] = df['integers'].astype(int)
df['floats'] = df['floats'].astype(float)
```

```
[12]: df
```

```
[12]:    integers  floats
      0         1     1.1
      1         2     2.2
      2         3     3.3
```

```
[13]: df.dtypes
```

```
[13]: integers    int32
      floats     float64
      dtype: object
```

Categorical Data:

- ▶ **Nominal:** Categories without a specific order (e.g., color: red, green, blue).
- ▶ **Ordinal:** Categories with a specific order (e.g., rating: low, medium, high).
- ▶ **Handling:** Encode categorical variables using techniques such as label encoding or one-hot encoding.

- ▶ **Label encoding:**
- ▶ **Label encoding** is a technique used to convert ordinal type of categorical data into numerical format.
- ▶ It assigns a unique integer to each category in the categorical variable.
- ▶ This is particularly useful for machine learning algorithms that require numeric input.
- ▶ `dataset['Col_Name'] = dataset[' Col_Name '].astype('category')`
- ▶ `dataset['Col_Name'] = dataset['Col_Name'].cat.codes`
- ▶ convert the `['Col_Name']` to categorical type using `astype('category')`.
- ▶ then use `cat.codes` to assign numeric labels to each category in the `['Col_Name']` and create a new column with the encoded values.

- ▶ **One-hot encoding:**
- ▶ **One-hot encoding** is a technique used to convert Nominal type categorical data into a binary format, where each category is represented as a binary vector.
- ▶ It creates new binary columns (also known as dummy variables) for each category, with a value of 1 indicating the presence of that category and 0 indicating absence.
- ▶ After OHE drop one variable. Here is python code:
- ▶ **dataset = pd.get_dummies(dataset, columns = ['Col_Name'])**

Where:

“**pd.get_dummies()**” is a pandas function used for one-hot encoding categorical variables.

“**columns=['Col_Name']**” specifies the column(s) in the Data Frame that you want to encode.

Categorical Data:

```
from sklearn.preprocessing import LabelEncoder, OneHotEncoder

df = pd.DataFrame({'category': ['red', 'green', 'blue']})
label_encoder = LabelEncoder()
df['category_encoded'] = label_encoder.fit_transform(df['category'])

one_hot_encoder = OneHotEncoder(sparse=False)
df_one_hot = pd.DataFrame(one_hot_encoder.fit_transform(df[['category']]))

columns=one_hot_encoder.categories_)
```

```
[15]: df_one_hot
```

```
[15]:   blue  green  red
      0    0.0    0.0  1.0
      1    0.0    1.0  0.0
      2    1.0    0.0  0.0
```

```
[16]: df_one_hot.dtypes
```

```
[16]: blue      float64
      green     float64
      red      float64
      dtype: object
      dtype: object
```

```
[17]: df['category_encoded']
```

```
[17]:  0    2
      1    1
      2    0
Name: category_encoded, dtype: int32
```

Datetime Data:

```
df = pd.DataFrame({'date': ['2020-01-01', '2021-01-01']})  
df['date'] = pd.to_datetime(df['date'])  
df['year'] = df['date'].dt.year  
df['month'] = df['date'].dt.month  
df['day'] = df['date'].dt.day
```

```
[22]: df['date']
```

```
[22]: 0    2020-01-01  
      1    2021-01-01  
      Name: date, dtype: datetime64[ns]
```

```
[23]: df['year']
```

```
[23]: 0    2020  
      1    2021  
      Name: year, dtype: int32
```

```
[24]: df['month']
```

```
[24]: 0    1  
      1    1  
      Name: month, dtype: int32
```

```
[25]: df['day']
```

```
[25]: 0    1  
      1    1  
      Name: day, dtype: int32
```

Convert Data Types:

```
df['integer'] = df['integer'].astype(int)
df['float'] = df['float'].astype(float)
df['category'] = df['category'].astype('category')
df['date'] = pd.to_datetime(df['date'])
```

Example

```
[59]: df['num_numerical'] = pd.to_numeric(df['number'], errors='coerce', downcast='integer')
```

```
[60]: df.head()
```

```
[60]:
```

	Cabin	Ticket	number	Survived	num_numerical
0	NaN	A/5 21171	5	0	5.0
1	C85	PC 17599	3	1	3.0
2	NaN	STON/O2. 3101282	6	1	6.0
3	C123	113803	3	1	3.0
4	NaN	373450	A	0	NaN

```
pd.to_numeric(df['number'], errors='coerce', downcast='integer')
```

- **pd.to_numeric**: This function is used to convert argument to a numeric type.
- **df['number']**: This specifies the column number from the DataFrame df that we want to convert.
- **errors='coerce'**: This argument tells the function how to handle errors during the conversion process. Specifically:**'coerce'**: This means that any values that cannot be converted to a numeric type will be set to NaN (Not a Number).
- **downcast='integer'**: This argument attempts to downcast the numeric type to the smallest possible integer subtype, which helps in saving memory.
 - For example, if all values can be represented by a smaller integer type (like int8), it will use that type instead of a larger one (like int64).

```
[61]: df['num_categorical'] = np.where(df['num_numerical'].isnull(), df['number'], np.nan)
```

```
[63]: df.head(20)
```

	Cabin	Ticket	number	Survived	num_numerical	num_categorical
0	NaN	A/5 21171	5	0	5.0	NaN
1	C85	PC 17599	3	1	3.0	NaN
2	NaN	STON/O2. 3101282	6	1	6.0	NaN
3	C123	113803	3	1	3.0	NaN
4	NaN	373450	A	0	NaN	A
5	NaN	330877	2	0	2.0	NaN
6	E46	17463	2	0	2.0	NaN

```
np.where(df['num_numerical'].isnull(), df['number'], np.nan)
```

1. np.where(condition, x, y):

- This function from the NumPy library is used for element-wise selection from two arrays (x and y) based on a condition. It returns elements chosen from x or y depending on the condition.
- condition: This specifies the condition to be checked.
- x: The values to select where the condition is True.
- y: The values to select where the condition is False.

2. df['num_numerical'].isnull():

- This checks for NaN values in the num_numerical column of the DataFrame df. It returns a boolean Series where True indicates the presence of a NaN value and False indicates the absence of a NaN value.

3. df['number']:

- This specifies the original number column from the DataFrame df, which contains the original values before any numeric conversion.

4. np.nan:

- This specifies that NaN should be used where the condition is False.

Data Balancing

- ▶ Importance of balanced datasets.
- ▶ Data is said to be imbalance if twice of minority class is less than the majority class.
- ▶ To find it, check the count in dependent variables.
- ▶ Two popular approach to solve the problem:
 - ▶ Oversampling
 - ▶ SMOTE

```
[10]: y.value_counts()  
[10]: 0    284315  
      1     492  
      Name: Class, dtype: int64
```

RandomOverSampler

- ▶ Simply duplicates random instances of the minority class to increase its representation in the dataset.
- ▶ This can be effective but may lead to overfitting as the same instances are repeated multiple times.

```
[ ]: #!pip install imblearn
import imblearn
# split the data into feature variable and Label/result/output variable
x = dataset.iloc[:, :-1]
y = dataset.iloc[:, -1]

[14]: from imblearn.over_sampling import RandomOverSampler
over = RandomOverSampler()
x_over, y_over = over.fit_resample(x,y)

[15]: y_over.value_counts()

[15]: 0    284315
      1    284315
      Name: Class, dtype: int64

[16]: y_over.shape

[16]: (568630,)
```

SMOTE

(Synthetic minority oversampling technique)

- Generates synthetic samples by interpolating between existing minority class instances.
- This technique creates more diverse samples compared to simple duplication, potentially reducing overfitting.

```
[17]: from imblearn.over_sampling import SMOTE  
smote = SMOTE()  
x_smote, y_smote = smote.fit_resample(x,y)  
y_smote.value_counts()
```

```
[17]: 0    284315  
1    284315  
Name: Class, dtype: int64
```

Cross Tabs for Feature Relationships

A cross tabulation presents the frequency distribution of variables in a matrix format. Each cell in the table represents the count or frequency of the occurrences of the specific combination of categories from the variables.

Why Use Cross Tabulation?

1. **Identify Relationships:** It helps in identifying relationships between categorical variables.
2. **Detect Patterns:** It can detect patterns and trends in the data.
3. **Summarize Data:** It provides a compact summary of data.
4. **Inform Decision-Making:** It aids in making informed decisions based on the observed relationships.

Example: Using Cross Tabulation in Python

Let's consider a dataset where we want to examine the relationship between two categorical variables: `Gender` and `Purchased`.

Sample Data:

plaintext

 Copy code

Gender	Purchased
Male	Yes
Female	No
Female	Yes
Male	No
Female	Yes
Male	Yes

Creating a Cross Tabulation:

1. Load the Data:

python

Copy code

```
import pandas as pd\n\ndata = {\n    'Gender': ['Male', 'Female', 'Female', 'Male', 'Female', 'Male'],\n    'Purchased': ['Yes', 'No', 'Yes', 'No', 'Yes', 'Yes']\n}\n\ndf = pd.DataFrame(data)
```

2. Create Cross Tab:

```
python
```

 Copy code

```
cross_tab = pd.crosstab(df['Gender'], df['Purchased'])  
print(cross_tab)
```

Output:

```
plaintext
```

 Copy code

Purchased	No	Yes
Gender		
Female	1	2
Male	1	2

3. Add Margins (to get totals):

python

 Copy code

```
cross_tab_with_totals = pd.crosstab(df['Gender'], df['Purchased'], margins=True)  
print(cross_tab_with_totals)
```

Output:

plaintext

 Copy code

Purchased	No	Yes	All
Gender			
Female	1	2	3
Male	1	2	3
All	2	4	6

Interpretation:

- The table shows that there are 3 females and 3 males in the dataset.
- Among the females, 2 have made a purchase, and 1 has not.
- Among the males, 2 have made a purchase, and 1 has not.
- The total number of purchases is 4, and the total number of non-purchases is 2.

Visualization:

To make the relationship more visually intuitive, you can plot the cross-tabulated data.

1. Bar Plot:

```
python Copy code
cross_tab.plot(kind='bar', stacked=True)
import matplotlib.pyplot as plt
plt.title('Purchase Frequency by Gender')
plt.xlabel('Gender')
plt.ylabel('Frequency')
plt.show()
```

2. Heatmap:

```
python Copy code
import seaborn as sns

sns.heatmap(cross_tab, annot=True, cmap="YlGnBu", cbar=False)
plt.title('Heatmap of Purchase Frequency by Gender')
plt.show()
```

Exploratory Data Analysis (EDA):

- ▶ **Definition:** Exploratory Data Analysis (EDA) is a critical step in the data analysis process. It involves examining and summarizing the main characteristics of a dataset, often using visual methods.
- ▶ Here are some key steps and techniques you can use during EDA
- ▶ **Techniques:**
 - ▶ **Data visualization:** Histograms, scatter plots, box plots.
 - ▶ **Summary statistics:** Mean, median, standard deviation.
 - ▶ **Outlier detection:** Identifying data points that deviate significantly from the rest of the dataset

What we can do in EDA

1. Understand the Data Structure

- Load the Data: Import the dataset and understand its structure.

```
python
```

 Copy code

```
import pandas as pd  
df = pd.read_csv('your_dataset.csv')  
df.head()  
df.info()  
df.describe()
```

2. Handling Missing Values

- Identify Missing Values: Check for missing values.

```
python
```

 Copy code

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

- Handle Missing Values: Depending on the context, you can drop or fill missing values.

```
python
```

 Copy code

```
df.dropna() # Drop missing values  
df.fillna(df.mean(), inplace=True) # Fill missing values with mean
```

3. Data Cleaning

- Remove Duplicates: Identify and remove duplicate rows.

python

 Copy code

```
df.drop_duplicates(inplace=True)
```

- Handle Outliers: Detect and treat outliers.

python

 Copy code

```
df.boxplot(column='column_name')
```

4. Data Transformation

- Feature Engineering: Create new features from existing ones.

python

 Copy code

```
df['new_feature'] = df['existing_feature1'] / df['existing_feature2']
```

- Encoding Categorical Variables: Convert categorical variables to numerical ones.

python

 Copy code

```
df = pd.get_dummies(df, columns=['categorical_column'])
```

5. Univariate Analysis

- **Summary Statistics:** Get descriptive statistics for individual features.

python

 Copy code

```
df['column_name'].describe()
```

- **Visualization:**

- **Histograms:**

python

 Copy code

```
df['column_name'].hist()
```

- **Box Plots:**

python

 Copy code

```
df.boxplot(column='column_name')
```

6. Bivariate Analysis

- Correlation: Check correlations between numerical features.

```
python
```

 Copy code

```
df.corr()
```

- Visualization:

- Scatter Plots:

```
python
```

 Copy code

```
df.plot.scatter(x='feature1', y='feature2')
```

- Pair Plots:

```
python
```

 Copy code

```
import seaborn as sns  
sns.pairplot(df)
```

7. Multivariate Analysis

- **Heatmaps:** Visualize correlation matrices.

```
python
```

 Copy code

```
sns.heatmap(df.corr(), annot=True)
```

- **Group By:** Group data and perform aggregations.

```
python
```

 Copy code

```
df.groupby('categorical_column').mean()
```

8. Distribution Analysis

- **Density Plots:** Examine the distribution of numerical features.

```
python
```

 Copy code

```
df['column_name'].plot(kind='density')
```

- **QQ Plots:** Assess if data follows a certain distribution.

```
python
```

 Copy code

```
import statsmodels.api as sm
import matplotlib.pyplot as plt
sm.qqplot(df['column_name'], line ='45')
plt.show()
```

9. Feature Relationships

- **Cross Tabs:** Examine relationships between categorical features.

python

 Copy code

```
pd.crosstab(df['categorical_feature1'], df['categorical_feature2'])
```

10. Dimensionality Reduction

- **PCA:** Apply Principal Component Analysis for high-dimensional data.

python

 Copy code

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
principalComponents = pca.fit_transform(df)
```

Visualization Libraries

- **Matplotlib:** Basic plotting.

```
python
```

 Copy code

```
import matplotlib.pyplot as plt  
plt.plot(df['column_name'])
```

- **Seaborn:** Advanced visualizations.

```
python
```

 Copy code

```
sns.boxplot(x='categorical_feature', y='numerical_feature', data=df)
```

- **Plotly:** Interactive plots.

```
python
```

 Copy code

```
import plotly.express as px  
fig = px.scatter(df, x='feature1', y='feature2')  
fig.show()
```

Visualization

- ▶ Using plots for making inferences in machine learning involves visualizing data to understand its structure, relationships, and patterns, which can guide feature selection, model choice, and evaluation.

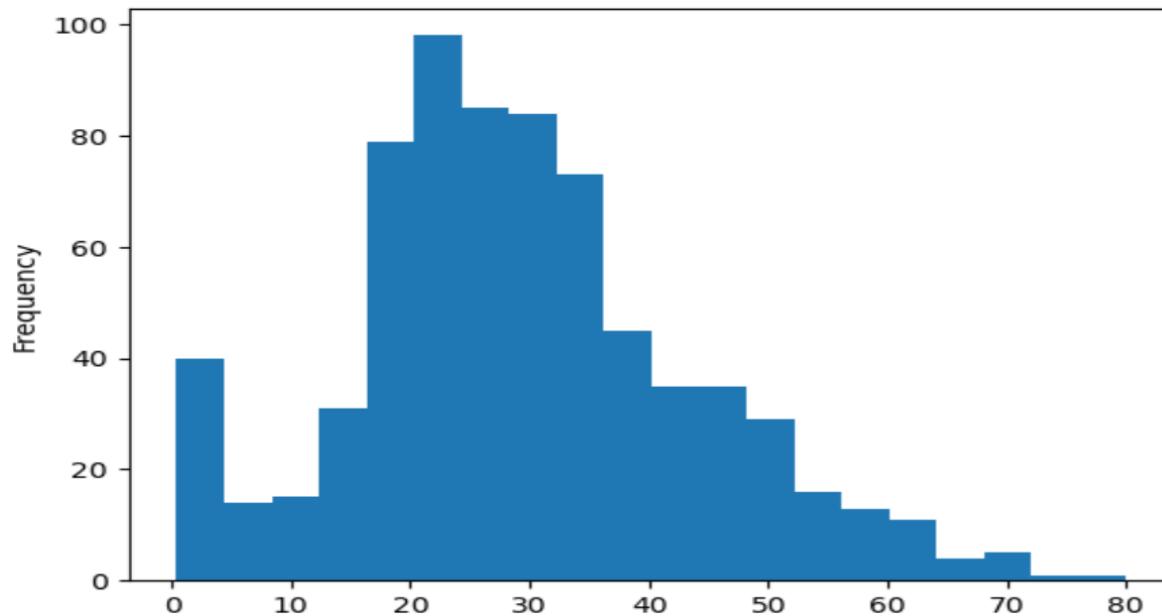
Visualization Techniques for EDA

- ▶ Histograms
- ▶ Box plots
- ▶ Scatter plots
- ▶ Pair plots
- ▶ Correlation matrix and heatmaps
- ▶ Bar plots
- ▶ Count plots
- ▶ Violin plots

Histograms

- ▶ **Purpose:** Understand the distribution of individual variables.
- ▶ **Inference:** Identify skewness, outliers, and the presence of multiple modes in the data.

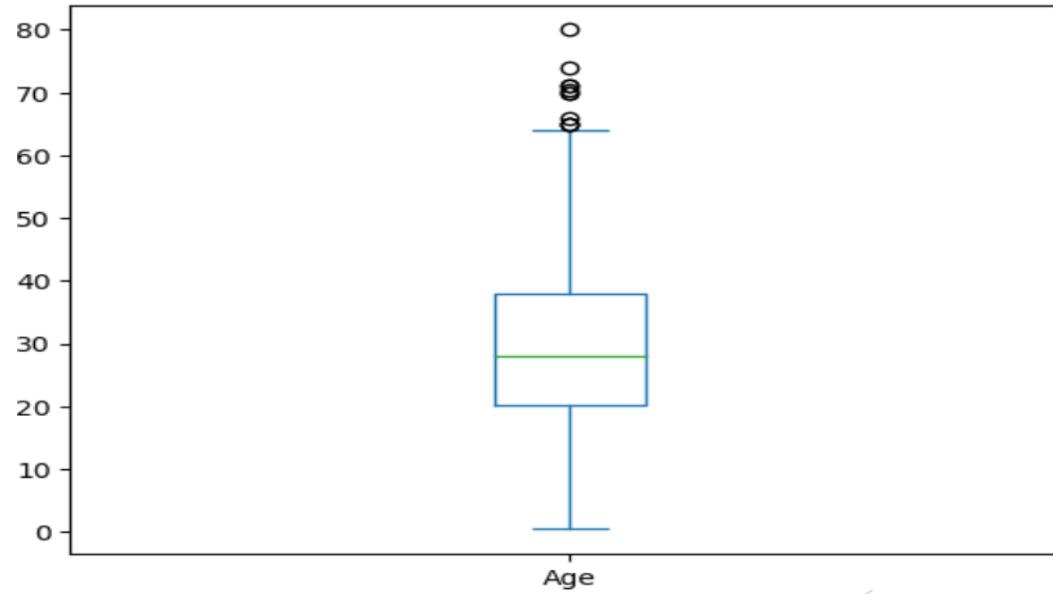
```
dataset['Age'].plot(kind='hist', bins=20)
```



Box Plots

- ▶ **Purpose:** Summarize the distribution of a dataset.
- ▶ **Inference:** Detect outliers, understand the spread and symmetry of the data.

```
dataset['Age'].plot(kind='box')
```

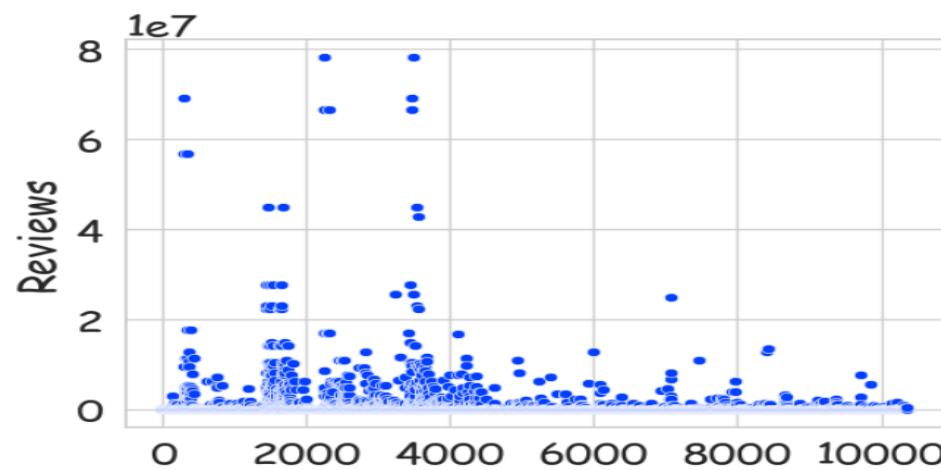


Scatter Plots

- ▶ **Purpose:** Visualize the relationship between two continuous variables.
- ▶ **Inference:** Identify correlations, clusters, and potential outliers

```
[142]: sns.set_theme(style='whitegrid', palette='bright', font='cursive', font_scale=1.8)
```

```
[148]: sns.scatterplot(y=df.Reviews, x=df.Reviews.index)
plt.show()
```



Pair Plots (Scatterplot Matrix)

- ▶ **Purpose:** Visualize pairwise relationships between multiple variables.
- ▶ **Inference:** Detect relationships between pairs of features, spot trends, clusters, and outliers.

Correlation Matrix and Heatmaps

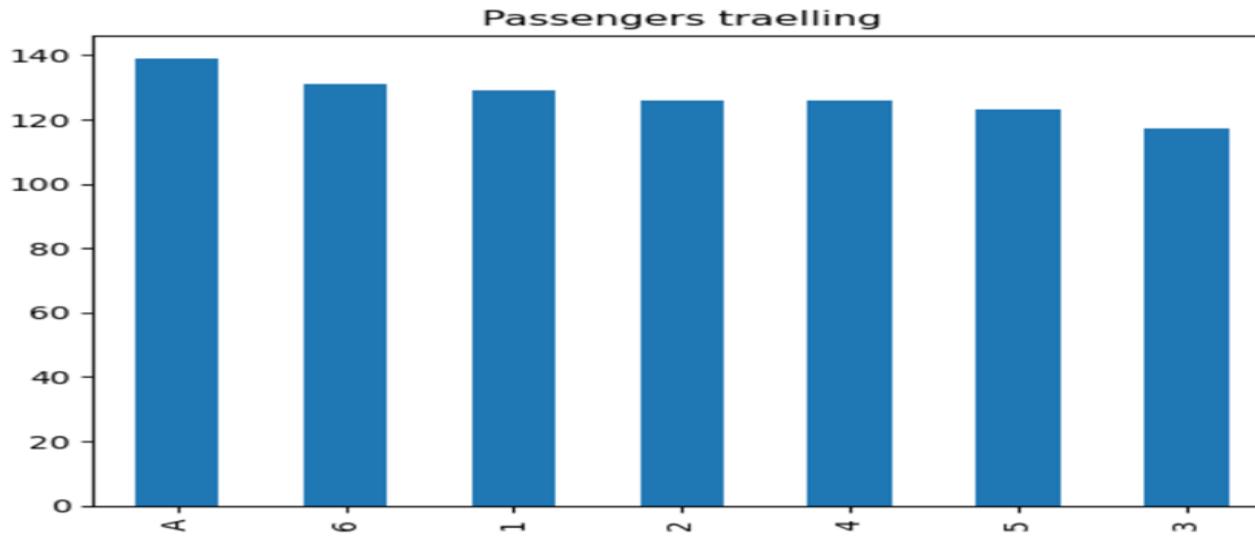
- ▶ **Purpose:** Show the correlation coefficients between variables.
- ▶ **Inference:** Identify highly correlated features that might be redundant.

```
plt.figure(figsize=(8, 6))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

Bar Plots

- ▶ **Purpose:** Compare categorical data.
- ▶ **Inference:** Understand the frequency distribution of categorical features.

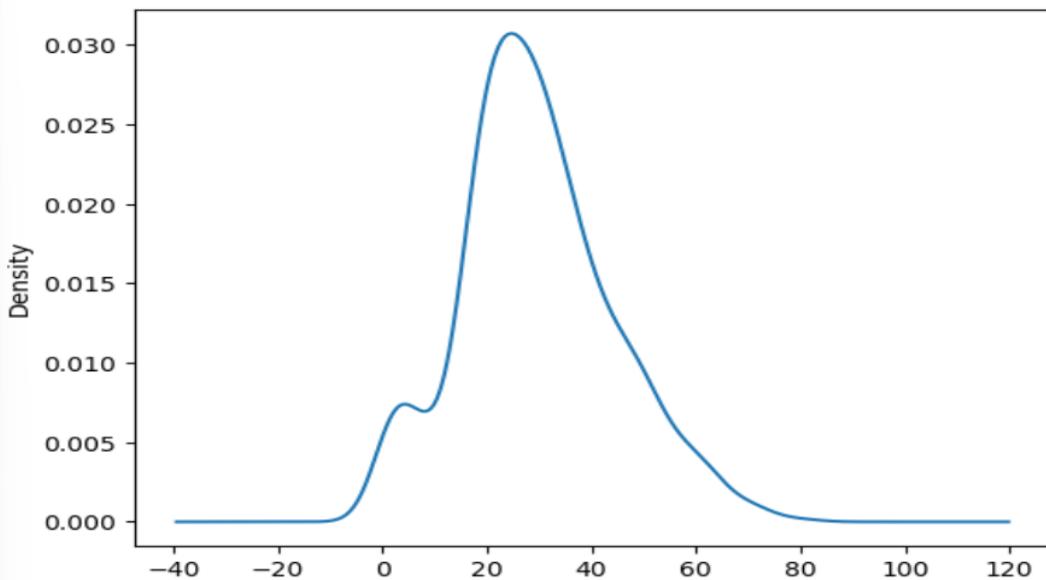
```
fig = df['number'].value_counts().plot.bar()  
fig.set_title("Passengers traelling")
```



Density plots

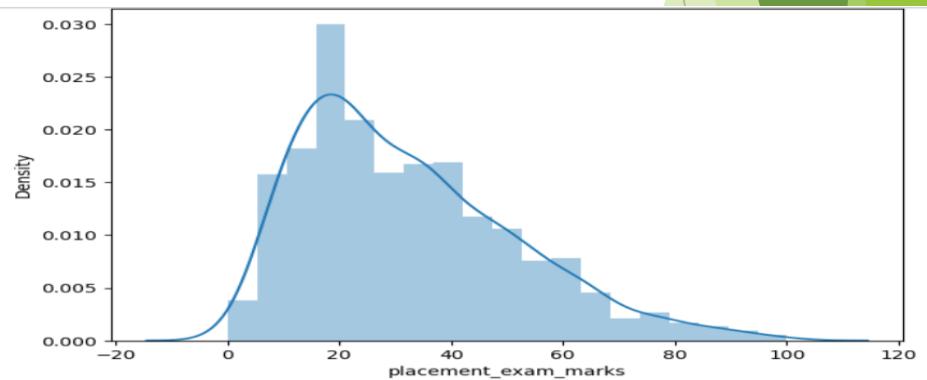
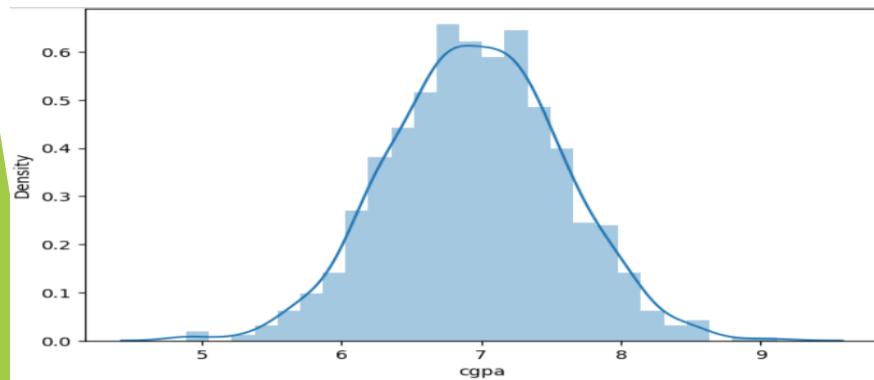
- ▶ Density plots display the probability density function of a continuous variable. They are useful for visualizing the overall shape of the distribution and comparing multiple distributions.

```
dataset['Age'].plot(kind='kde')
```



Distribution plot

- ▶ A **distribution plot** is a visualization that combines aspects of a histogram and a kernel density plot to show the distribution of a continuous variable.
- ▶ It is useful for understanding the distribution of data points in a dataset and identifying patterns such as skewness, kurtosis, and the presence of outliers.
- ▶ `sns.histplot` is used instead of `sns.distplot`.
- ▶ The parameter `kde=True` adds the KDE line to the histogram



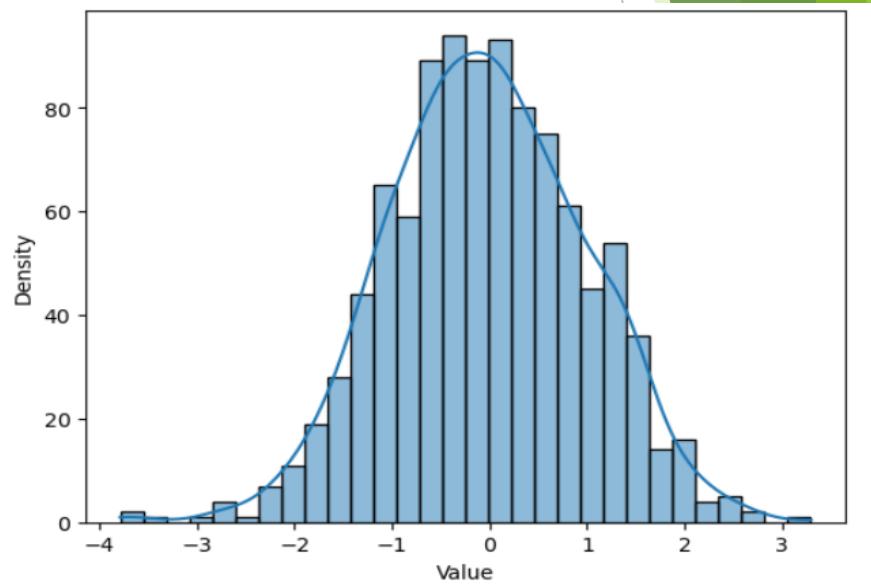
- In recent versions of Seaborn (0.11.0 and later), `sns.distplot` has been deprecated.
- Instead, you should use `sns.histplot` or `sns.kdeplot` for similar functionality. Here's how to create a similar plot using `sns.histplot`

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

# Example data
import numpy as np
data = np.random.randn(1000) # Generating random data

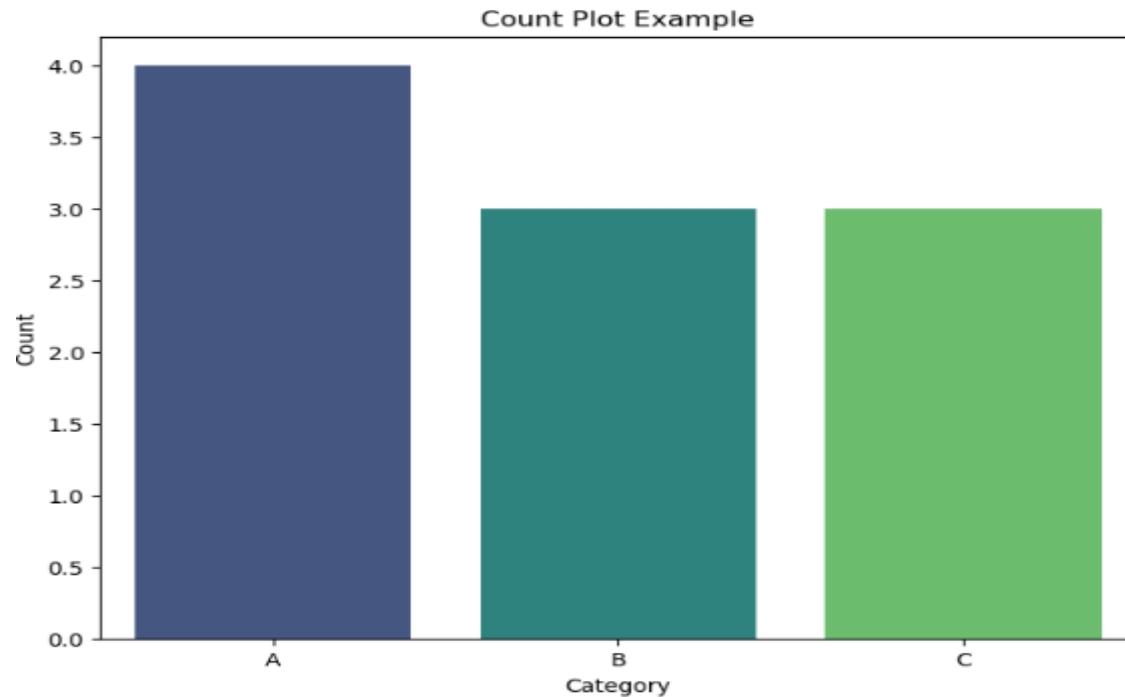
# Creating a histogram with KDE
sns.histplot(data, kde=True, bins=30)

# Adding Labels and title
plt.xlabel('Value')
plt.ylabel('Density')
```



Count Plots

- ▶ **Purpose:** Show the counts of observations in each categorical bin.
- ▶ **Inference:** Detect the distribution of categorical features.



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

# Sample data
import pandas as pd
data = pd.DataFrame({
    'Category': ['A', 'B', 'A', 'C', 'B', 'A', 'C', 'C', 'B', 'A']
})

# Create the count plot
plt.figure(figsize=(8, 6)) # Optional: Set figure size
sns.countplot(data=data, x='Category', palette='viridis')

# Add title and labels
plt.title('Count Plot Example')
plt.xlabel('Category')
plt.ylabel('Count')

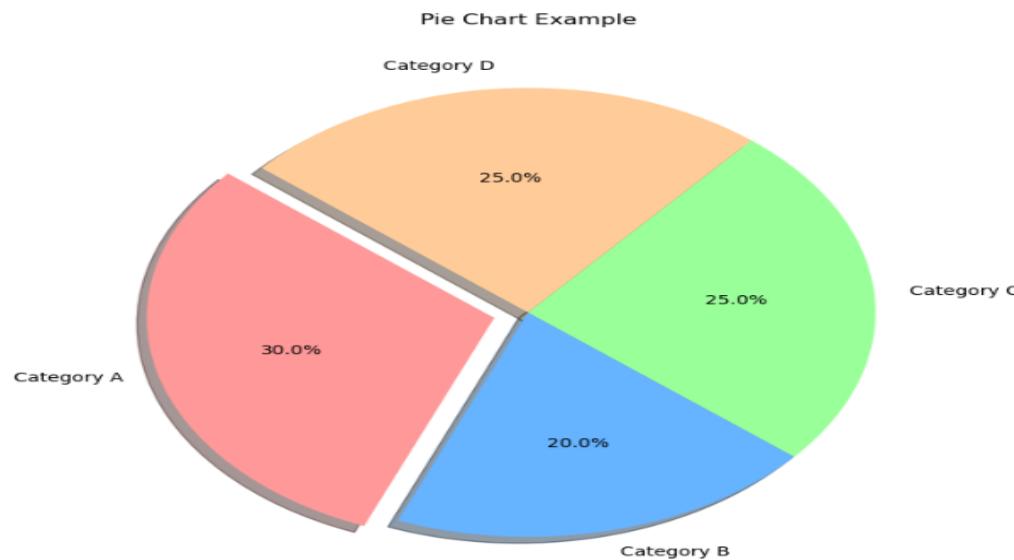
# Show plot
plt.show()
```

Explanation:

- `data`: The DataFrame containing the data.
- `x`: The column name containing the categorical data you want to plot.
- `palette`: Specifies the color palette for the bars. You can use predefined palettes like 'viridis', 'coolwarm', etc., or define your own colors.

Pie Plot

- ▶ A pie plot (or pie chart) is a circular statistical graphic that is divided into slices to illustrate numerical proportions. Each slice represents a category's proportion to the whole dataset. Pie charts are useful for showing the relative sizes of parts to a whole, making it easy to compare the parts of a single categorical variable.



```
import matplotlib.pyplot as plt

# Sample data
labels = ['Category A', 'Category B', 'Category C', 'Category D']
sizes = [30, 20, 25, 25] # Corresponding sizes of each category
colors = ['#ff9999','#66b3ff','#99ff99','#ffcc99'] # Optional: colors for each slice
explode = (0.1, 0, 0, 0) # Optional: explode the first slice for emphasis

# Create the pie chart
plt.figure(figsize=(8, 8)) # Optional: Set figure size
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%', shadow=True, startangle=140)

# Add title
plt.title('Pie Chart Example')

# Show plot
plt.show()
```

Explanation:

- `labels`: The names of the categories to be shown in the pie chart.
- `sizes`: The values corresponding to each category.
- `colors`: The colors for each slice of the pie chart. This is optional.
- `explode`: A tuple to indicate which slice to "explode" (or pull out) for emphasis. The first value `0.1` means the first slice will be slightly pulled out.
- `autopct`: A string format for the percentage labels on each slice.
- `shadow`: Adds a shadow effect to the pie chart.
- `startangle`: Rotates the pie chart to start from a specified angle.

Feature Elimination Method

- ▶ If number of features are too large to handle than it is wise approach to remove insignificant features.
- ▶ There two popular approach.
- ▶ PCA :Principal Component Analysis
- ▶ RFE: Recursive Feature Elimination

Recursive Feature Elimination (RFE)

- ▶ **Purpose:**
- ▶ RFE is a feature selection method that iteratively removes less important features based on the model's performance, identifying the most influential features for predicting the target variable.
- **Mathematical Basis:**
 1. **Feature Importance:**
 - RFE uses an estimator (e.g., Logistic Regression, Random Forest, etc.) that provides feature importance, such as weights or coefficients.

Steps:

- Train the model on the dataset.
- Rank features based on their importance scores.
- Remove the least important feature(s).
- Repeat until the desired number of features remains.

➤ **Advantages:**

- Identifies the most critical features for the model.
- Helps improve model performance by eliminating redundant or irrelevant features.
- Works well with small to medium datasets.

➤ **Disadvantages:**

- Computationally expensive for large datasets.
- Performance depends on the chosen estimator.

Code Example:

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
random_state=42)

# Logistic Regression as estimator
logitR = LogisticRegression()

# Apply RFE to select top 2 features
selector = RFE(estimator=logitR, n_features_to_select=2, step=1)
selector.fit(X_train, y_train)

# Selected features
print("Selected Features:", selector.support_)
```

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split

# Load dataset
data = load_iris()
X, y = data.data, data.target

# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Logistic Regression as estimator
logitR = LogisticRegression()

# Apply RFE to select top 2 features
selector = RFE(estimator=logitR, n_features_to_select=2, step=1)
selector.fit(X_train, y_train)

# Selected features
print("Selected Features:", selector.support_)
print("Feature Ranking:", selector.ranking_)
```

Principal Component Analysis (PCA)

- ▶ **Purpose:**
- ▶ PCA is a dimensionality reduction technique that transforms the dataset into a lower-dimensional space while preserving as much variance as possible.

Mathematical Basis:

1. Covariance Matrix:

- Compute the covariance matrix of the dataset.

$$\text{Covariance Matrix: } \Sigma = \frac{1}{n-1} \sum_{i=1}^n (X_i - \mu)(X_i - \mu)^T$$

2. Eigenvalues and Eigenvectors:

- Calculate eigenvalues and eigenvectors of the covariance matrix.
- Eigenvalues represent the variance captured by each principal component.
- Eigenvectors represent the directions of maximum variance.

3. Principal Components:

- Transform the original dataset X into a new coordinate system using the eigenvectors.

$$Z = X \cdot W$$

Where W is the matrix of eigenvectors.

Advantages:

- Reduces dimensionality while retaining variance.
- Removes multicollinearity by creating uncorrelated components.
- Improves computational efficiency for large datasets.

Disadvantages:

- Components are linear combinations of original features, losing interpretability.
- Sensitive to scaling; features must be normalized.

Key Differences

Aspect	RFE	PCA
Purpose	Feature Selection	Dimensionality Reduction
Method	Eliminates less important features iteratively	Transforms data into new components
Model Dependence	Model-dependent	Model-independent
Interpretability	Retains original feature meaning	Loses interpretability
Computational Cost	Higher (iterative process)	Lower for fewer components

Data Exploration & Data Preprocessing

2 Chapter

Data Objects

- Data sets are made up of data objects.
- A **data object** represents an entity.
- Examples:
 - sales database: customers, store items, sales
 - medical database: patients, treatments
 - university database: students, professors, courses
- Also called *samples* , *examples*, *instances*, *data points*, *objects*, *tuples*.
- Data objects are described by **attributes**.
- Database rows -> data objects; columns -> attributes.

Attributes

- **Attribute (or dimensions, features, variables):**
 - a data field, representing a characteristic or feature of a data object.
 - *E.g., customer_ID, name, address*
- Types:
 - Nominal
 - Binary
 - Ordinal
 - Numeric
 - Interval-scaled
 - Ratio-scaled

Attribute Types

1) Nominal Attribute :

- Related to names
- categories, states, or “names of things”
 - *Hair_color = {black, blond, brown, grey, red, white}*
 - marital status, occupation, ID numbers
- Also called as categorical Attribute

2) Binary Attribute

- Nominal attribute with only 2 states (0 and 1)
- 0 means attributes absent and 1 means attribute present
- Also called Boolean if two state corresponds to true and false
- Symmetric binary: both outcomes equally important and carry same weight.
 - e.g., gender
- Asymmetric binary: outcomes not equally important
 - e.g., medical test (positive vs. negative)
 - Convention: assign 1 to most important outcome (e.g., HIV positive)

- **Ordinal Attribute**
 - Values have a meaningful order (ranking) but magnitude between successive values is not known.
 - $Size = \{small, medium, large\}$,
 - grades=(A+, A, B, B+)

Numeric Attribute Types

Numeric Attribute:

- Quantitative (integer or real-valued)
- **Interval-scaled attributes**
 - Measured on a scale of **equal-sized units**
 - Values have order
 - E.g., *temperature in C° or F°, calendar dates*
 - **No true zero-point**
- **Ratio-scaled attributes**
 - **Inherent zero-point**
 - We can speak of values as being a multiple of another value
(10 Kg is twice of 5 Kg).
 - e.g. *length, counts, monetary quantities*

Discrete vs. Continuous Attributes

- **Discrete Attribute**
 - Has only a finite or countably infinite set of values
 - E.g., zip codes, profession, or the set of words in a collection of documents
 - Sometimes, represented as **integer variables**
 - Note: Binary attributes are a special case of discrete attributes
- **Continuous Attribute**
 - Has **real numbers** as attribute values
 - E.g., temperature, height, or weight
 - Continuous attributes are typically represented as floating-point variables

Basic Statistical Descriptions of Data

- For successful data preprocessing it is essential to have overall picture of your data
- Statistical Descriptions can be used to identify properties of the data
- Three areas of basic SD of data-----

1) Measuring the central tendency of data:

- mean, median, mode (where most of the values fall?)

2) Measuring the dispersion of data:

Range, quartiles, variance, standard deviation, interquartile range, five number summary, box plot

3) Graphic display of basic sd of data:

Quantile plot, quantile quantile plot, histogram, scatter plots

- Mean:
 - Effective measure of the center of a set of data
 - Let x_1, x_2, \dots, x_n be set of n values for numeric attribute X like salary. The mean is given by---

$$\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i \quad \text{Mean} = (x_1+x_2+\dots+x_n)/n$$

- Ex. 30,36,47,50,52,52,56,60,63,70,70,110

- Weighted arithmetic mean

$$\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$

- Trimmed mean: chopping extreme values

Median:

1. For **odd number** of values (count), Middle value is considered.
2. For **even number** of values (count), average of two middle most values is considered.
3. For **interval---**

$$\text{median} = L_1 + \left(\frac{n/2 - (\sum \text{freq})_l}{\text{freq}_{\text{median}}} \right) \text{width}$$

age	frequency
1–5	200
6–15	450
16–20	300
21–50	1500
51–80	700
81–110	44

Median
interval

Here

L1 → lower boundary of the median interval

N → number of values in entire dataset

$(\sum \text{freq})_l$ → sum of frequencies of all of the intervals
that are lower than the median interval

Freq_{median} → frequency of the median interval

Width → width of the median interval

- $L_1 \rightarrow 20$
- $N \rightarrow 3194$
- $(\sum freq)_1 \rightarrow 950$
- $Freq_{median} \rightarrow 1500$
- $Width \rightarrow 30$

Ans- \rightarrow

Median=32.94 years

- Mode
 - A mode is defined as the value that has a higher frequency in a given set of values.
 - It is the value that appears the most number of times.
 - **Example:** In the given set of data: 2, 4, 5, 5, 6, 7, the mode of the data set is 5 since it has appeared in the set twice.

Bimodal, Trimodal & Multimodal (More than one mode)

- When there are two modes in a data set, then the set is called **bimodal**
 - For example, The mode of Set A = {2,2,2,3,4,4,5,5,5} is 2 and 5, because both 2 and 5 is repeated three times in the given set.
- When there are three modes in a data set, then the set is called **trimodal**
 - For example, the mode of set A = {2,2,2,3,4,4,5,5,5,7,8,8,8} is 2, 5 and 8
- When there are four or more modes in a data set, then the set is called **multimodal**
- Empirical formula for unimodal numeric data---

$$\text{mean} - \text{mode} = 3 \times (\text{mean} - \text{median})$$

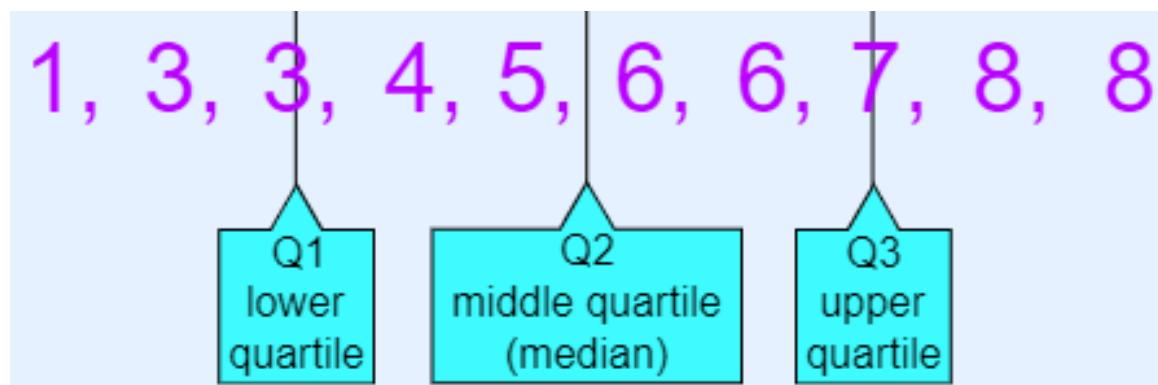
- **Midrange**
 - Average of largest and smallest value in dataset
 - $\text{Midrange} = (\text{Maximum Value} + \text{Minimum Value}) / 2$
 - **Example:** Consider the data set 110, 150, 180, 220, 270, 290, 310 and 390 as the prices of speakers. The minimum number is 110, and the maximum is 390.
 - $\text{Midrange} = (390 + 110) / 2 = 250$

Measuring the Dispersion of Data

- **Quartiles, outliers and boxplots:**
 - **Quartiles:** Q_1 (25^{th} percentile), Q_3 (75^{th} percentile)
 - **Inter-quartile range:** $\text{IQR} = Q_3 - Q_1$
 - **Five number summary:** $\{\text{min}, Q_1, \text{median}, Q_3, \text{max}\}$
 - **Boxplot:** ends of the box are the quartiles; median is marked; add whiskers, and plot outliers individually
 - **Outlier:** usually, a value higher/lower than $1.5 \times \text{IQR}$

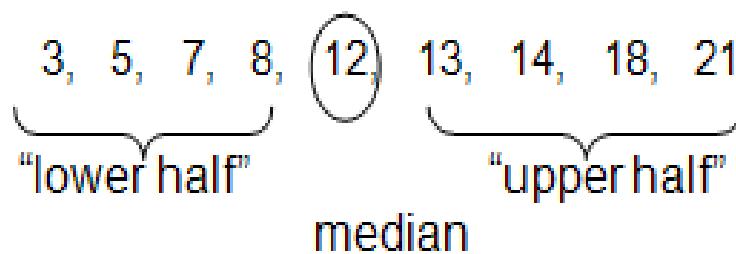
Quartiles

- Quartiles are the values (Q1, Q2,Q3,Q4) that divide a list of numbers into quarters or four parts.



Quartiles

- **Example 1:** Find the first and third quartiles of the data set $\{3, 7, 8, 5, 12, 14, 21, 13, 18\}$.
- **Total numbers in set=9**
- First, write data in increasing order: $3, 5, 7, 8, 12, 13, 14, 18, 21$.
- The median is 12.
- The first quartile, Q_1 , is the median of $\{3, 5, 7, 8\}=6$
- The third quartile, Q_3 , is the median of $\{13, 14, 18, 21\}=16$



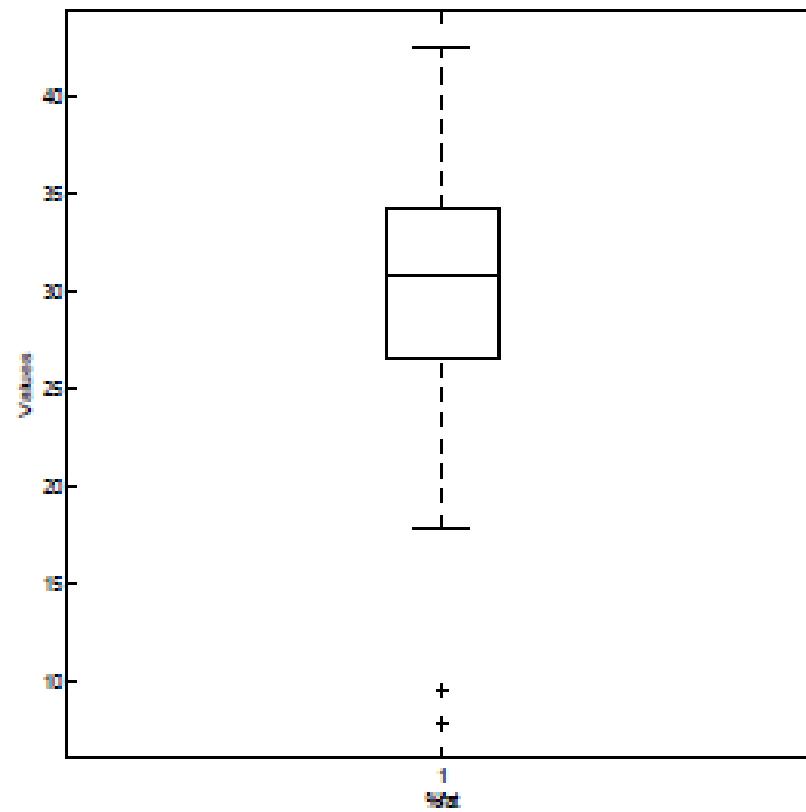
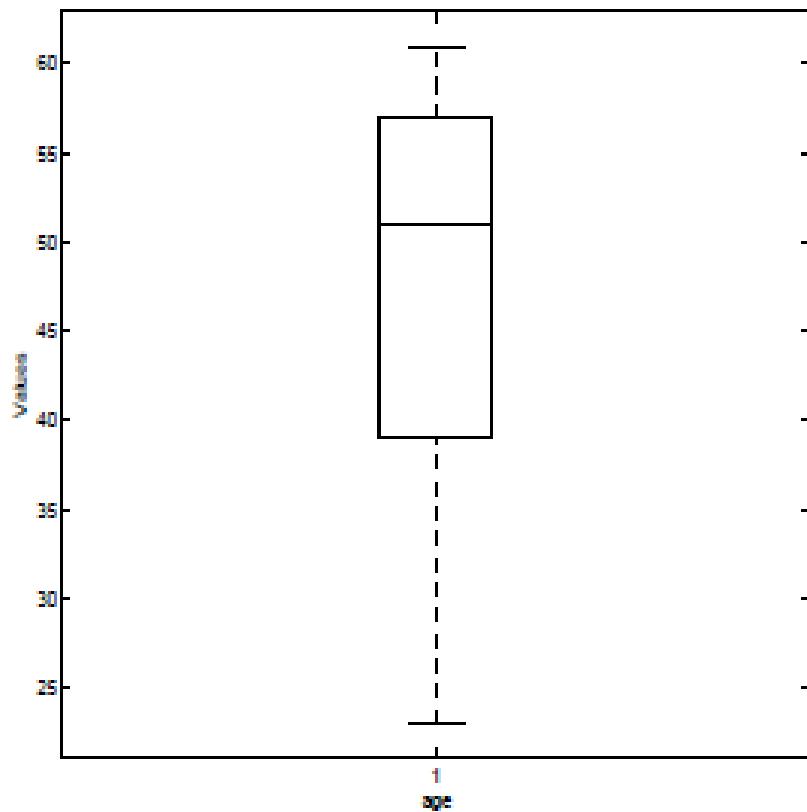
$$Q_1 = \frac{5+7}{2} = \frac{12}{2} = 6$$

$$Q_3 = \frac{14+18}{2} = \frac{32}{2} = 16$$

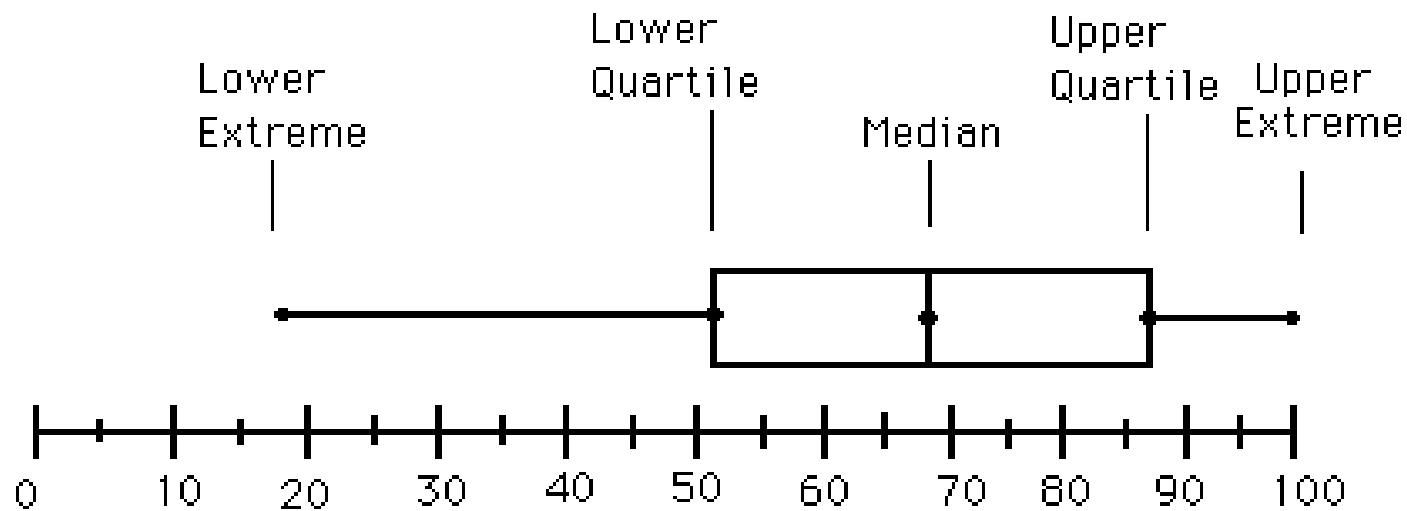
Quartiles

- **Example 2:** Find the first and third quartiles of the set {3, 7, 8, 5, 12, 14, 21, 15, 18, 14}.
- Median (Q2) is 13 (it is the mean of 12 and 14)
- $Q_1 = 8$
- $Q_3 = 15$.

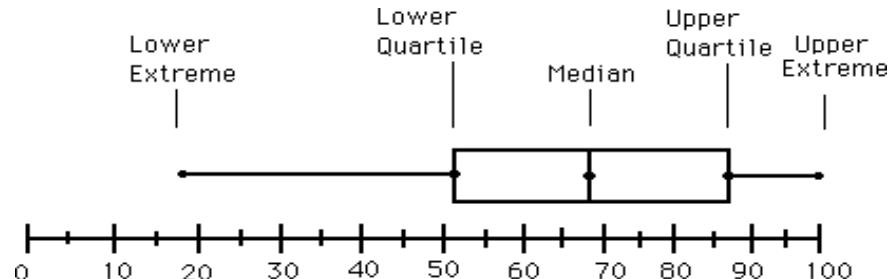
Boxplot



Boxplot



Boxplot Analysis



- **Five-number summary** of a distribution
 - Minimum, Q1, Median, Q3, Maximum
- **Boxplot**
 - Data is represented with a box
 - The ends of the box are at the first and third quartiles, i.e., the height of the box is IQR
 - The median is marked by a line within the box
 - Whiskers: two lines outside the box extended to Minimum and Maximum
 - Outliers: points beyond a specified outlier threshold, plotted individually

Boxplot Example 1

Example: A sample of 10 boxes of raisins has these weights (in grams):
25, 28, 29, 29, 30, 34, 35, 35, 37, 38

Make a box plot of the data

Solution:

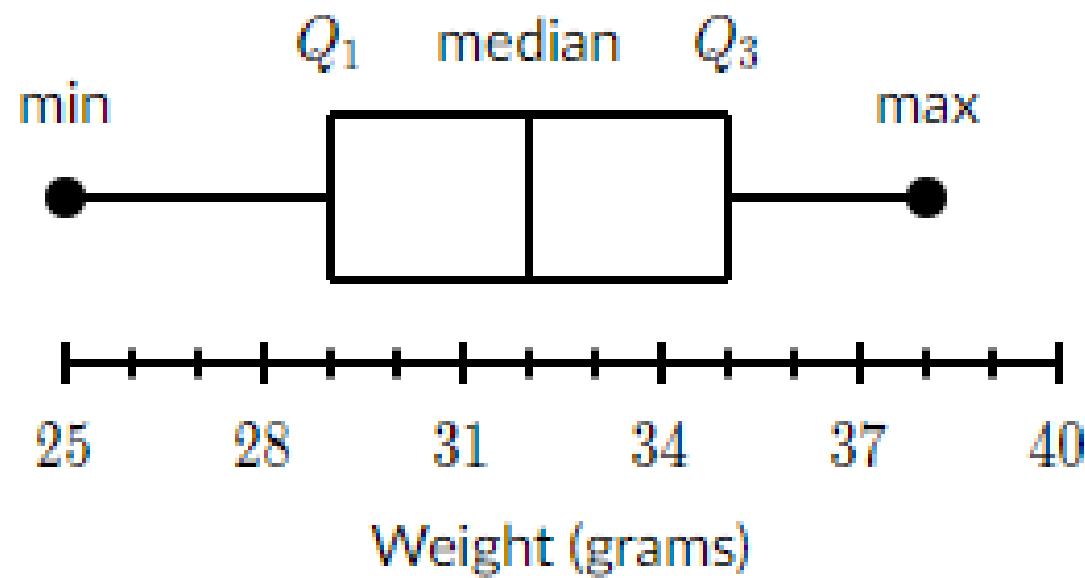
Step 1: Order the data from smallest to largest.

Step 2: find the 5 number summary

(minimum, first quartile, median, third quartile, and maximum)

→ (min, Q1,Q2,Q3,max)

→ (25,29,32,35,38)

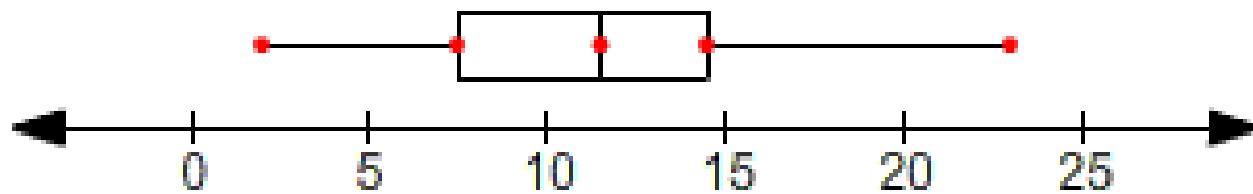


Five number summary: (25,29,32,35,38)

Ex 2

- Find Q1 , Q2 , and Q3 for the following data set, and draw a box-and-whisker plot.
- {2,6,7,8,8,11,12,13,14,15,22,23}

- Five number summary
- (2, 7.5, 11.5, 14.5, 23)



Boxplot Example 3

Ex 2. 30,36,47,50,52,52,56,60,63,70,70,110

- Draw the box plot

Ex 4

Find Q_1 , Q_2 , and Q_3 for the following data set. Identify any outliers, and draw a box-and-whisker plot.

$$\{5, 40, 42, 46, 48, 49, 50, 50, 52, 53, 55, 56, 58, 75, 102\}$$

There are 15 values, arranged in increasing order. So, Q_2 is the 8th data point, 50.

Q_1 is the 4th data point, 46, and Q_3 is the 12th data point, 56.

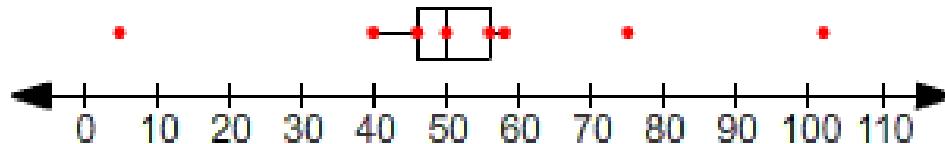
The interquartile range IQR is $Q_3 - Q_1$ or $56 - 46 = 10$.

Now we need to find whether there are values less than $Q_1 - (1.5 \times \text{IQR})$ or greater than $Q_3 + (1.5 \times \text{IQR})$.

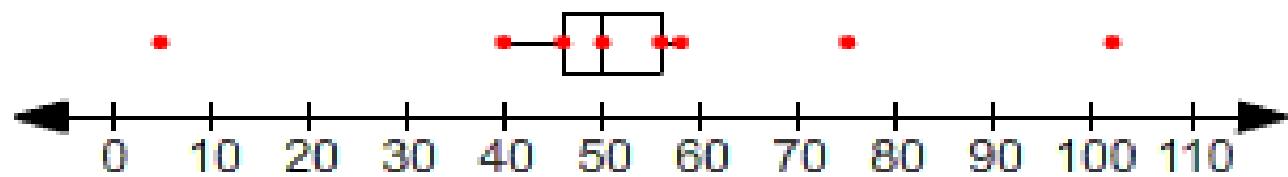
$$Q_1 - (1.5 \times \text{IQR}) = 46 - 15 = 31$$

$$Q_3 + (1.5 \times \text{IQR}) = 56 + 15 = 71$$

Since 5 is less than 31 and 75 and 102 are greater than 71, there are 3 outliers.



Note that 40 and 58 are shown as the ends of the whiskers with outliers plotted separately as a dots.



Outliers

- If a data value is very far away from the quartiles (either much less than Q1 or much greater than Q3), it is sometimes designated an outlier.
- The standard definition for an outlier is a number which is less than Q1 or greater than Q3 by more than 1.5 times the interquartile range
- $IQR = Q3 - Q1$
- That is, an outlier is any number less than $Q1 - (1.5 \times IQR)$ **or** greater than $Q3 + (1.5 \times IQR)$

Variance & Standard Deviation

- Variance: **Variance** is the sum of squares of differences between all numbers and means.

$$Formula : \sigma^2 = \frac{\sum_{i=1}^N (x_i - \mu)^2}{N}$$

- Where μ is Mean, N is the total number of elements or frequency of distribution.
- **Standard Deviation** is square root of variance. It is a measure of the extent to which data varies from the mean.
- The Standard Deviation is a measure of how spread out numbers are.

Example 1 – Standard deviation

A hen lays eight eggs. Each egg was weighed and recorded as follows

60 g, 56 g, 61 g, 68 g, 51 g, 53 g, 69 g, 54 g.

a. First, calculate the mean:

$$\begin{aligned}\bar{x} &= \frac{\sum x}{n} \\ &= \frac{472}{8} \\ &= 59\end{aligned}$$

*

b. Now, find the standard deviation.

Table 1. Weight of eggs, in grams

Weight (x)	(x - \bar{x})	$(x - \bar{x})^2$
60	1	1
56	-3	9
61	2	4
68	9	81
51	-8	64
53	-6	36
69	10	100
54	-5	25
472		320

Using the information from the above table, we can see that

$$\sum (x - \bar{X})^2 = 320$$

In order to calculate the standard deviation, we must use the following formula:

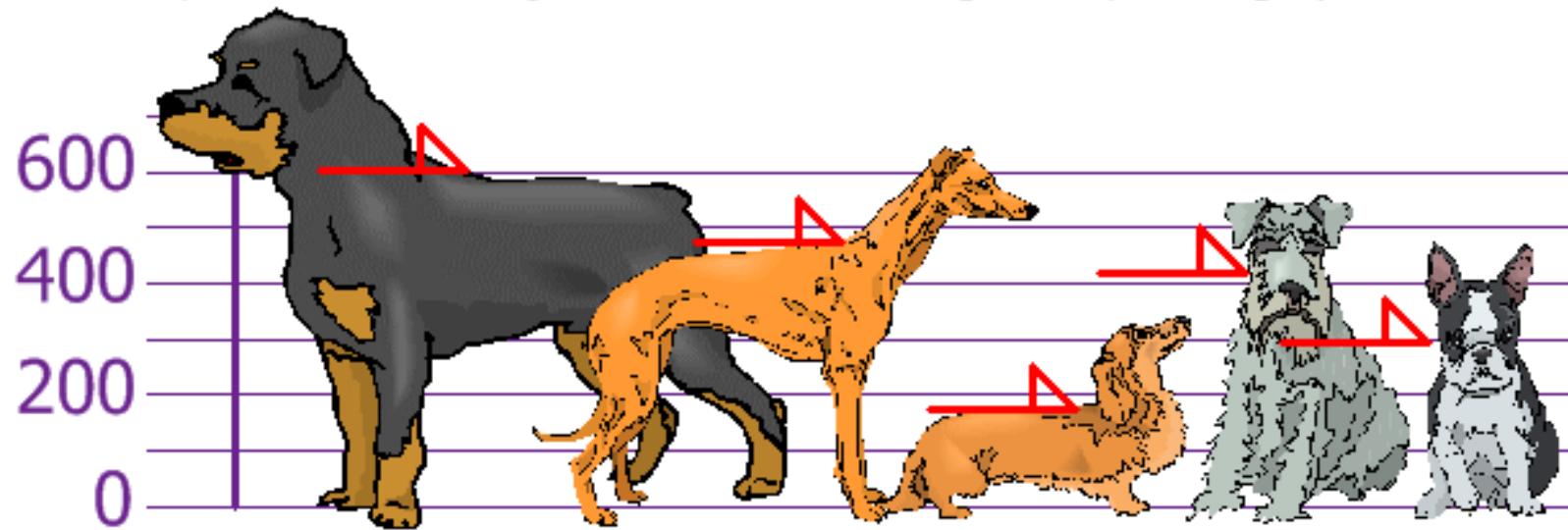
$$s = \sqrt{\frac{\sum (x - \bar{x})^2}{n}}$$

$$= \sqrt{\frac{320}{8}}$$

$$= 6.32 \text{ grams}$$

Example

You and your friends have just measured the heights of your dogs (in millimeters):



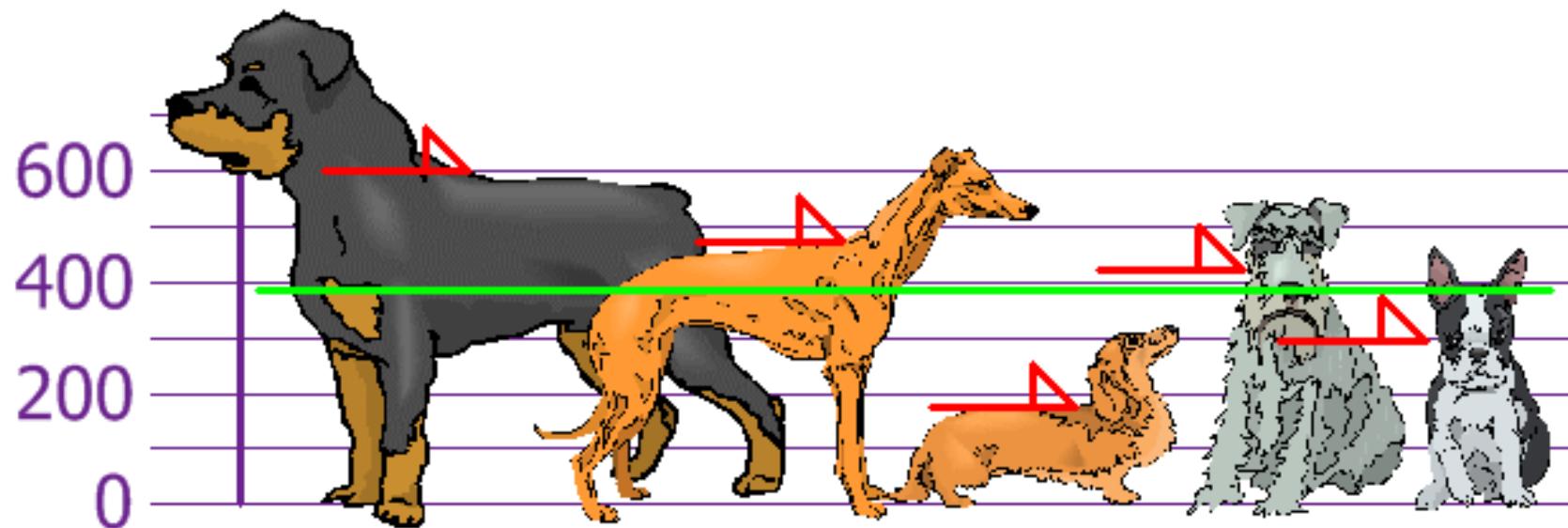
The heights (at the shoulders) are: 600mm, 470mm, 170mm, 430mm and 300mm.

Find out the Mean, the Variance, and the Standard Deviation.

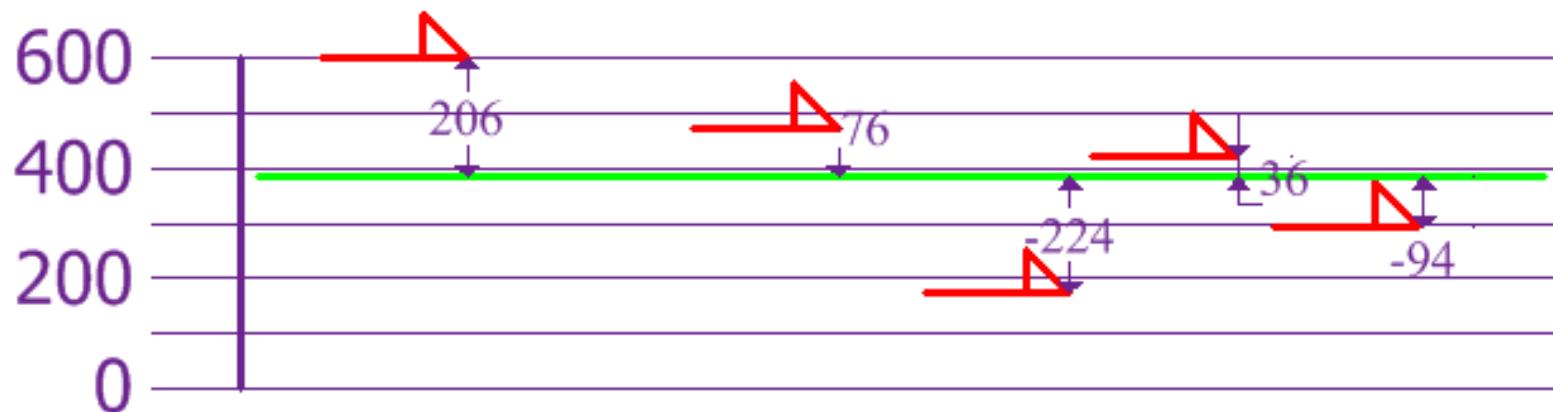
Answer:

$$\begin{aligned}\text{Mean} &= \frac{600 + 470 + 170 + 430 + 300}{5} \\ &= \frac{1970}{5} \\ &= 394\end{aligned}$$

so the mean (average) height is 394 mm. Let's plot this on the chart:



Now we calculate each dog's difference from the Mean:



To calculate the Variance, take each difference, square it, and then average the result:

Variance

$$\begin{aligned}\sigma^2 &= \frac{206^2 + 76^2 + (-224)^2 + 36^2 + (-94)^2}{5} \\&= \frac{42436 + 5776 + 50176 + 1296 + 8836}{5} \\&= \frac{108520}{5} \\&= 21704\end{aligned}$$

So the Variance is **21,704**

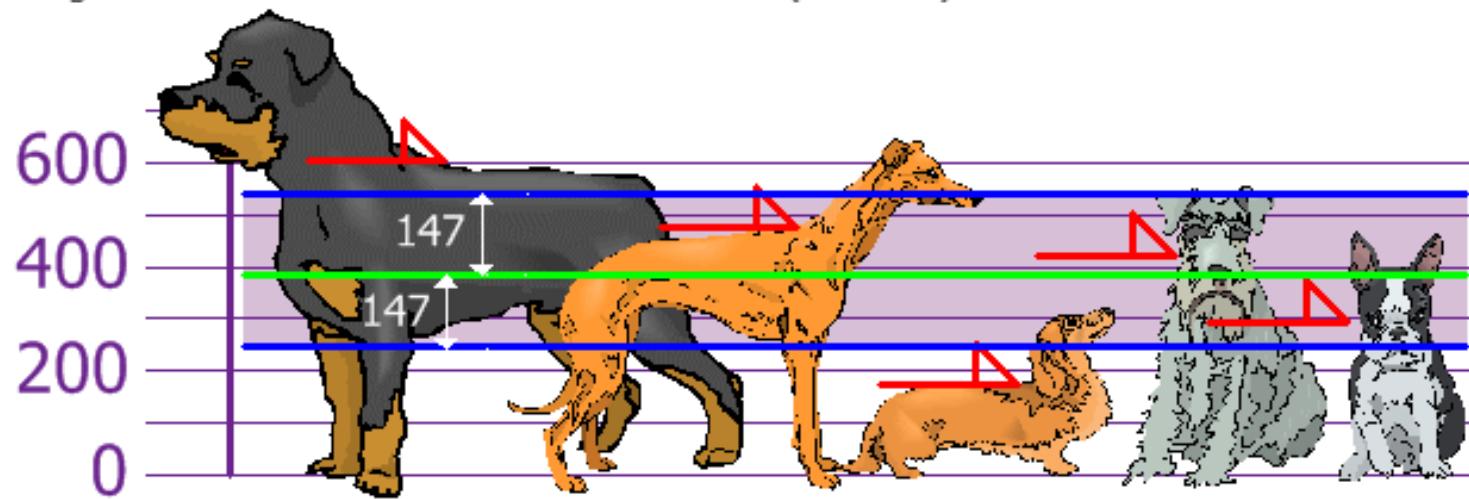
And the Standard Deviation is just the square root of Variance, so:

Standard Deviation

$$\begin{aligned}\sigma &= \sqrt{21704} \\ &= 147.32... \\ &= \mathbf{147} \text{ (to the nearest mm)}\end{aligned}$$

And the good thing about the Standard Deviation is that it is useful. Now we can show which heights are within one Standard Deviation (147mm) of the Mean:

And the good thing about the Standard Deviation is that it is useful. Now we can show which heights are within one Standard Deviation (147mm) of the Mean:



So, using the Standard Deviation we have a "standard" way of knowing what is normal, and what is extra large or extra small.

Rottweilers **are** tall dogs. And Dachshunds **are** a bit short, right?

Key points about variance/standard deviation

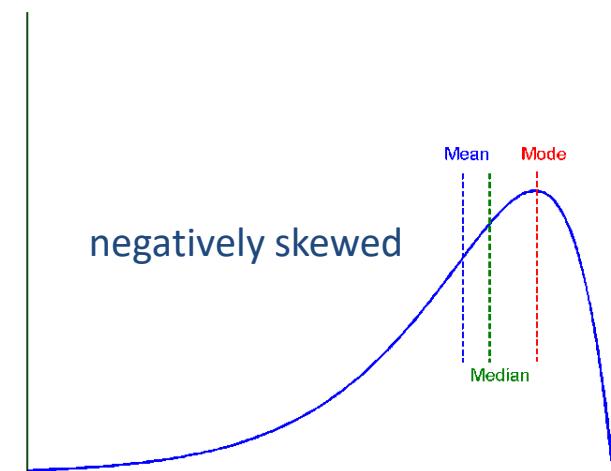
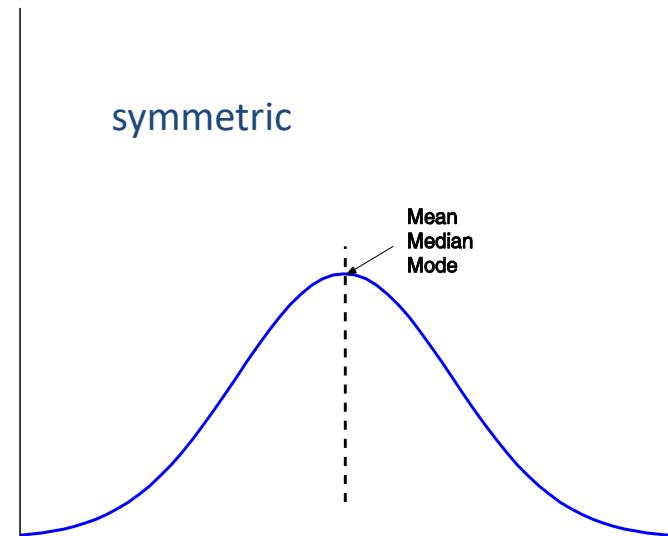
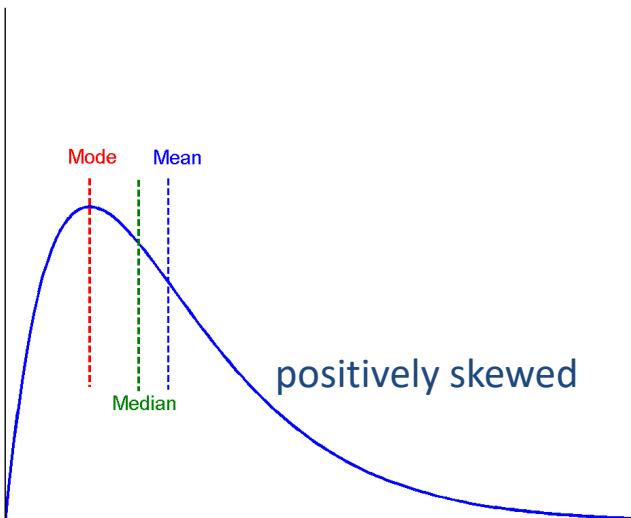
- Variance and standard deviation are measures of data dispersion
- They indicate how spread out a data distribution is.
- A **low standard deviation** means that the data observation tend to be very close to mean
- **High standard deviation** indicates data are spread out over a large range of values
- $\sigma = 0$ when there is no spread, that is , when all observation have the same value. Otherwise $\sigma > 0$

Symmetric vs. Skewed Data

- Median, mean and mode of symmetric, positively and negatively skewed data

"skewed to the left" (the long tail is on the left hand side): negatively skewed

"skewed to the right" (the long tail is on the right hand side): positively skewed

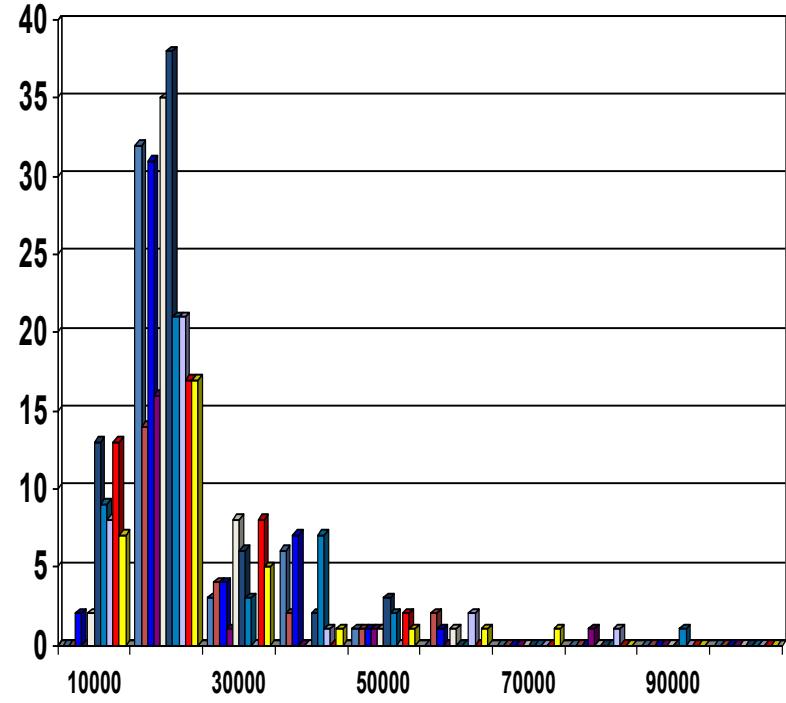


Graphic Displays of Basic Statistical Descriptions

- **Histogram:** x-axis are values, y-axis represents frequencies
- **Quantile plot:** each value x_i is paired with f_i indicating that approximately $100 f_i \%$ of data are $\leq x_i$
- **Quantile-quantile (q-q) plot:** graphs the quantiles of one univariant distribution against the corresponding quantiles of another
- **Scatter plot:** each pair of values is a pair of coordinates and plotted as points in the plane

Histogram Analysis

- Histogram: Graph display of tabulated frequencies, shown as bars
- It shows what proportion of cases fall into each of several categories
- Differs from a bar chart in that it is the *area* of the bar that denotes the value, not the height as in bar charts, a crucial distinction when the categories are not of uniform width
- The categories are usually specified as non-overlapping intervals of some variable. The categories (bars) must be adjacent



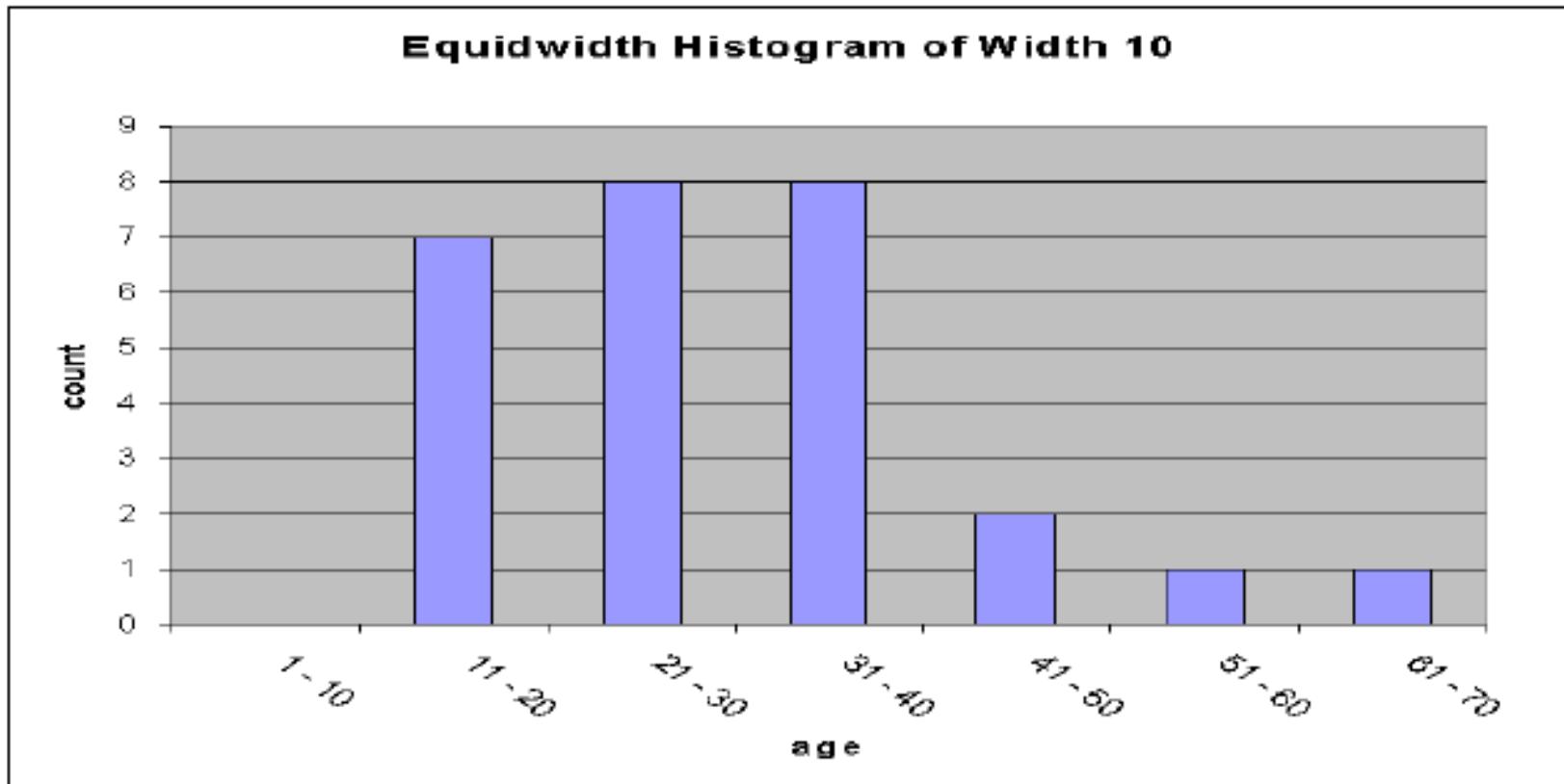
Example Histogram

- Dataset for Age:

13,15,16,16,20,20,21,22,25,25,25,25,25,30,33,33,35,35,
35,36,40,45,46,52,70

Age	Count/Frequency
13	1
15	1
16	2
20	2
21	1
22	1
25	4
30	1
33	2
35	4

Example Equidwidth Histogram



Example Histogram

Unit price(\$)	Count of items sold
40	275
43	300
47	250
74	360
75	515
78	540
115	320
117	270
120	350

Quantile plot

- Used to check whether your data is Normal
- To make a QQplot:

For a sample of size n : x_1, x_2, \dots, x_n

1. Order the data from smallest to largest:

$x_{(1)}, x_{(2)}, \dots, x_{(n)}$ where $x_{(i)}$ is the i -th smallest

2. Calculate the sample quantile

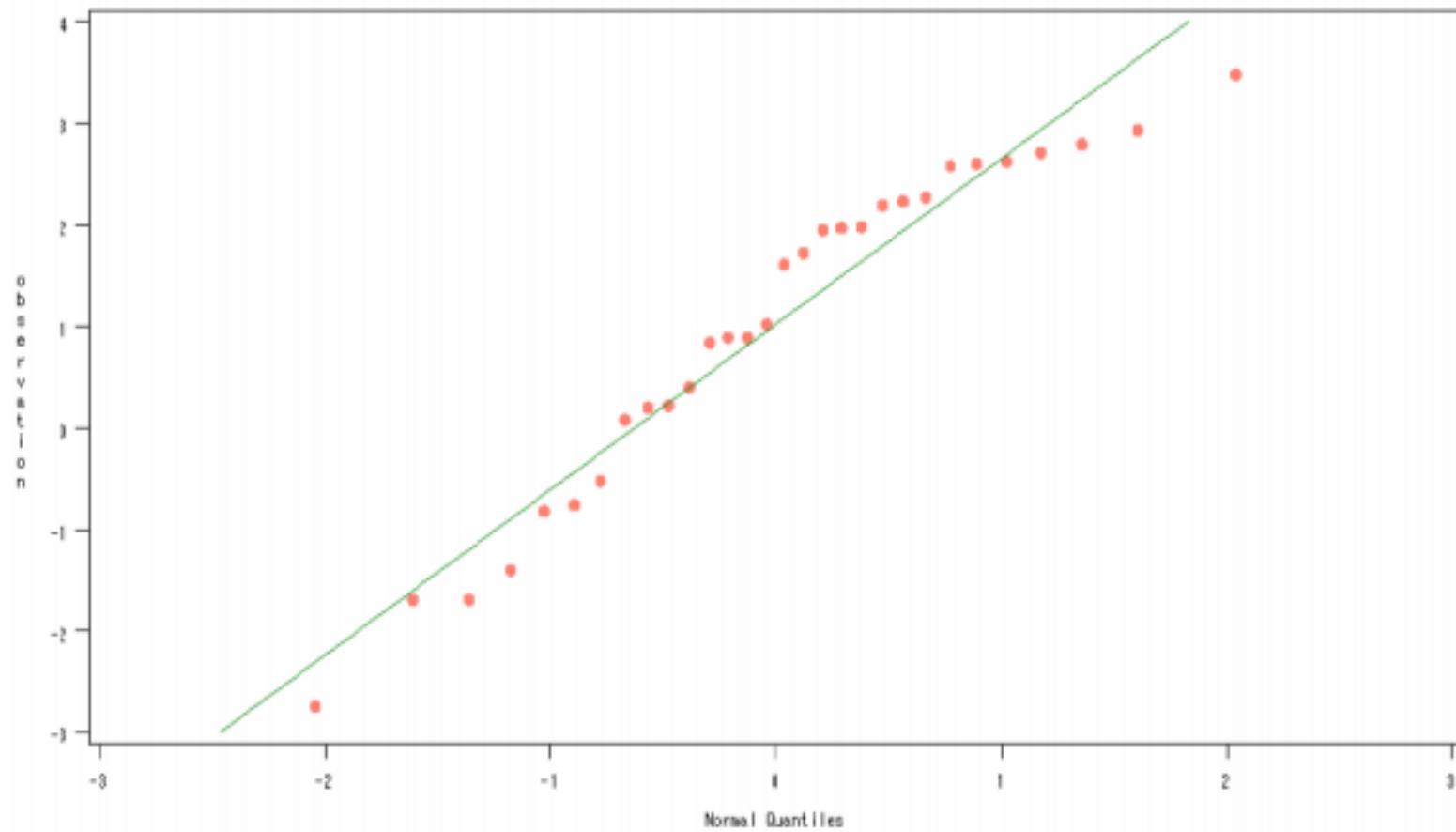
Sample quantile is calculated as:

$$x_{(i)} = [(i - 0.5)/n]\text{th sample quantile}$$

3. Plot the points $(([(i - 0.5)/n]\text{th z-percentile}, x_{(i)})$

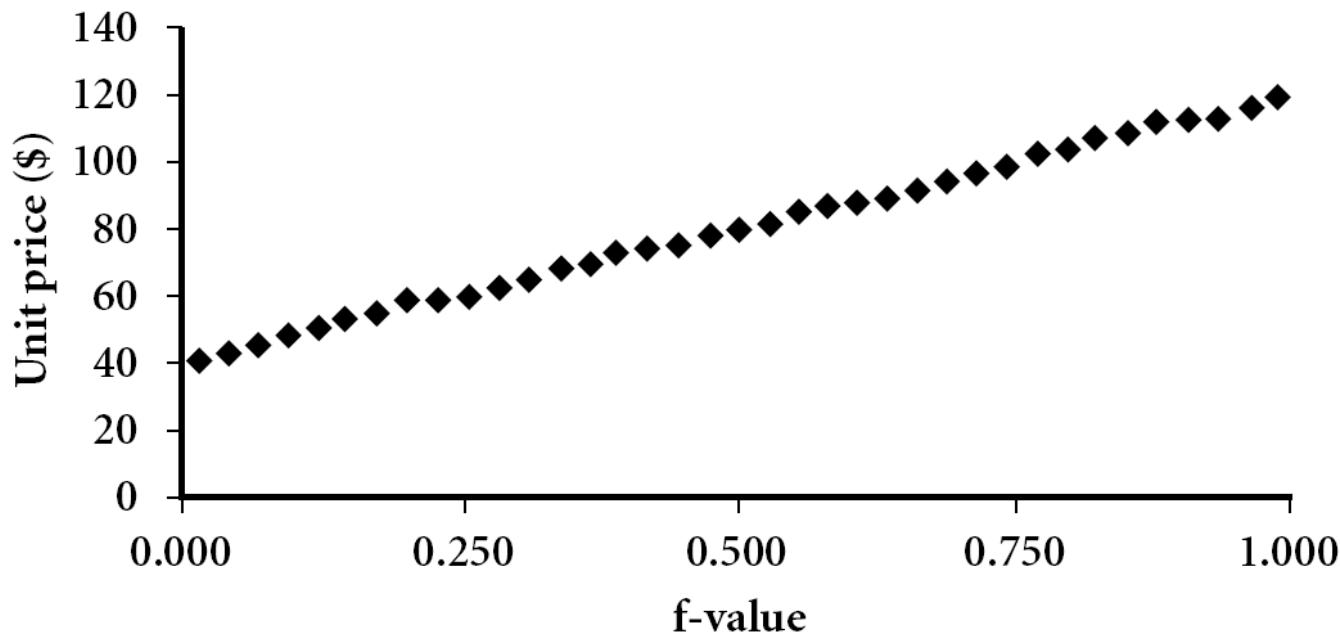
- If the data distribution is close to normal, the plotted points will lie close to a sloped straight line on the QQplot!

Quantile plot



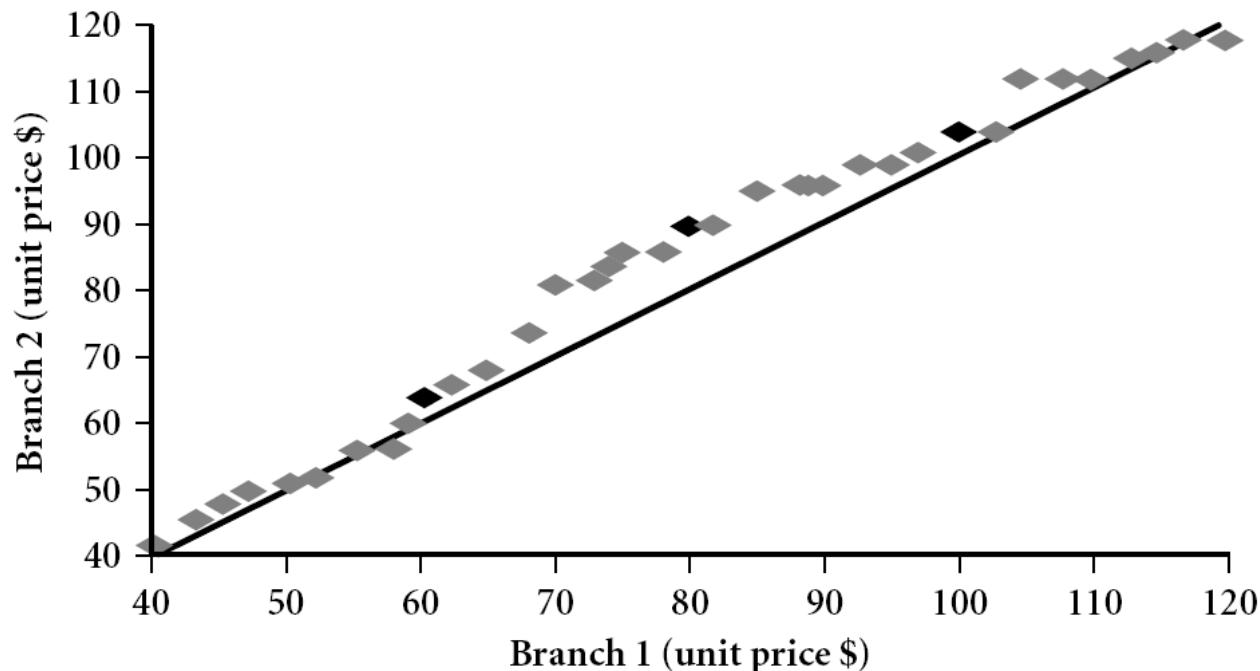
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots **quantile** information
 - For a data x_i , data sorted in increasing order, f_i indicates that approximately $100 f_i\%$ of the data are below or equal to the value x_i



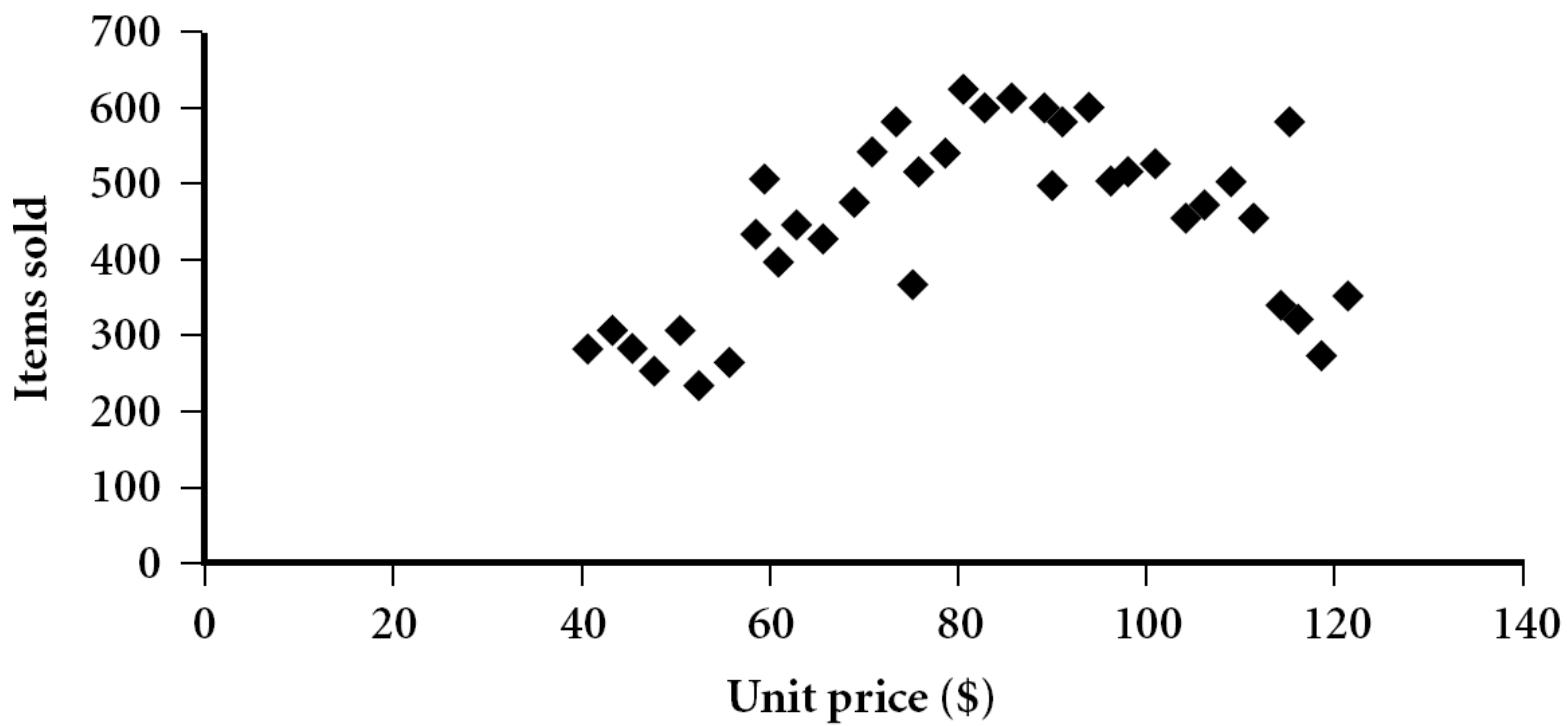
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- View: Is there is a shift in going from one distribution to another?
- Example shows unit price of items sold at Branch 1 vs. Branch 2 for each quantile. Unit prices of items sold at Branch 1 tend to be lower than those at Branch 2.

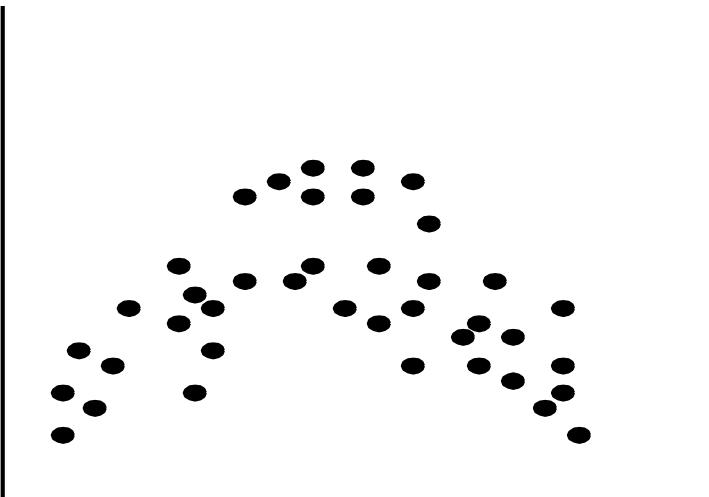
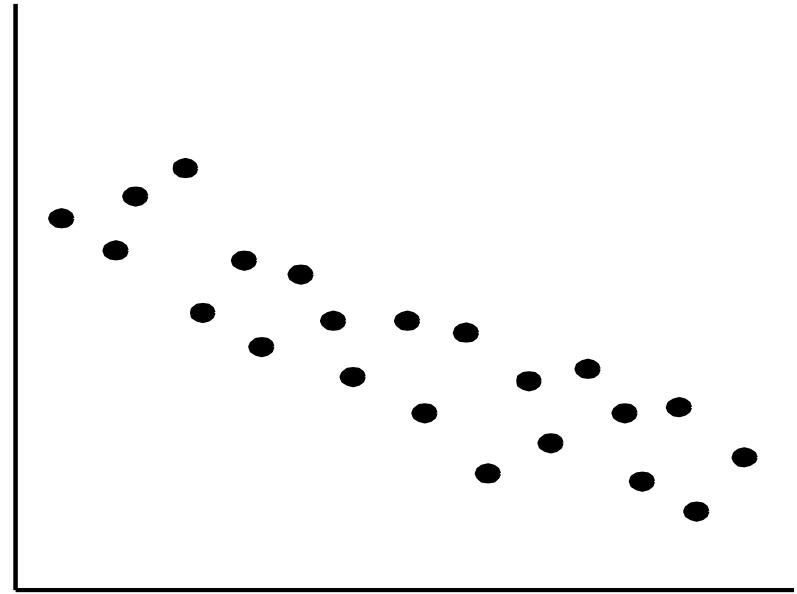
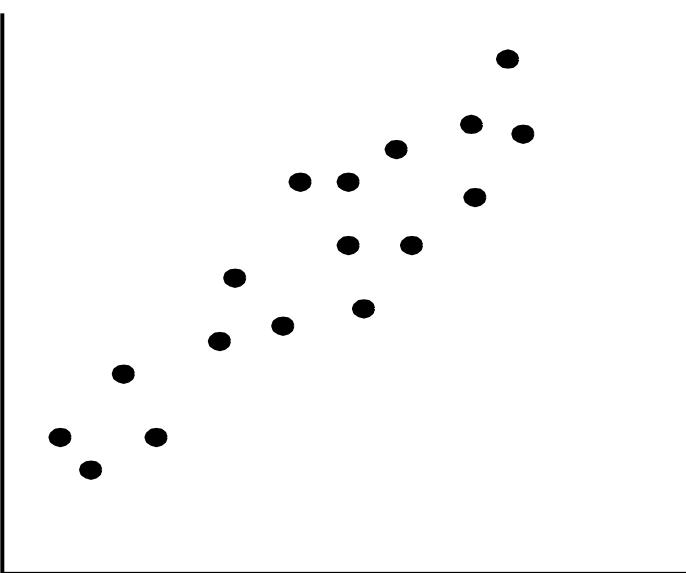


Scatter plot

- Each pair of values is treated as a pair of coordinates and plotted as points in the plane

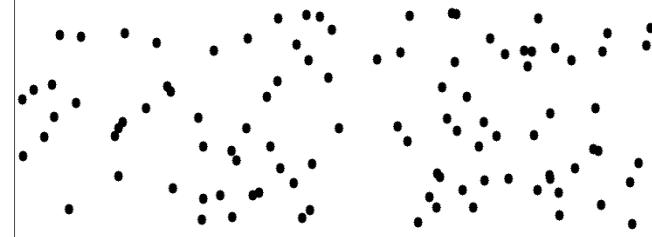
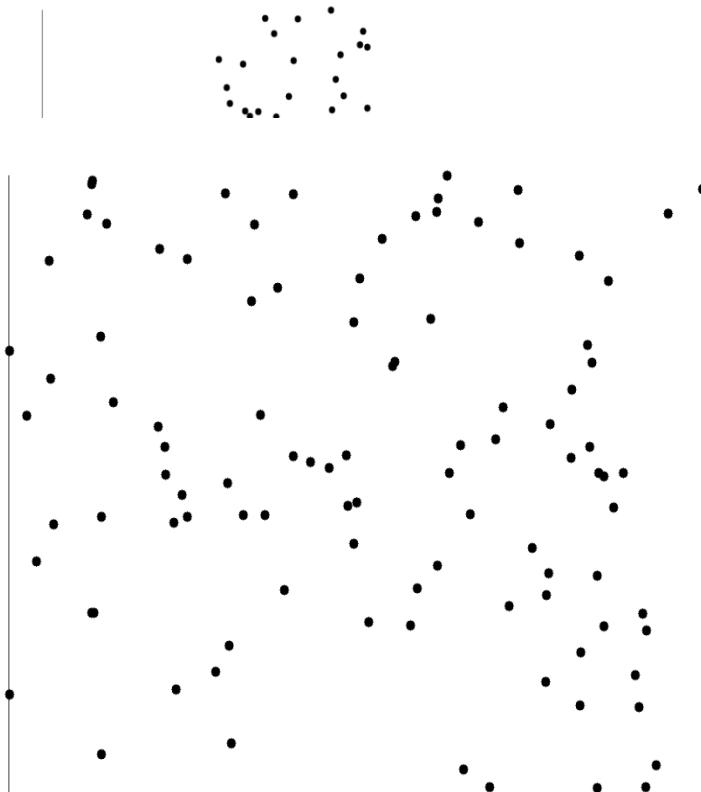


Positively and Negatively Correlated Data



- The left half fragment is positively correlated
- The right half is negative correlated

Uncorrelat

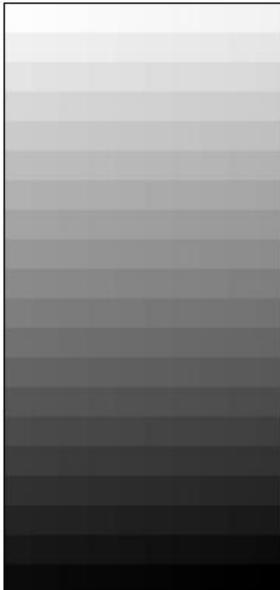


Data Visualization

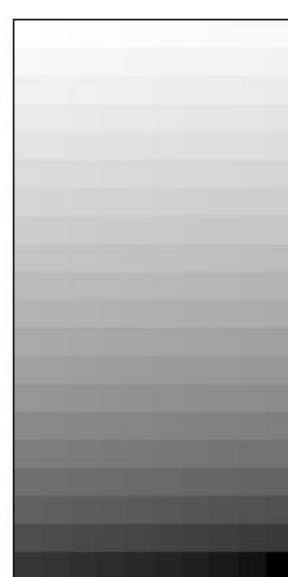
- Why data visualization?
 - Gain insight into an information space by mapping data onto graphical primitives
 - Provide qualitative overview of large data sets
 - Search for patterns, trends, structure, irregularities, relationships among data
 - Help find interesting regions and suitable parameters for further quantitative analysis
 - Provide a visual proof of computer representations derived
- Categorization of visualization methods:
 - Pixel-oriented visualization techniques
 - Geometric projection visualization techniques
 - Icon-based visualization techniques
 - Hierarchical visualization techniques
 - Visualizing complex data and relations

Pixel-Oriented Visualization Techniques

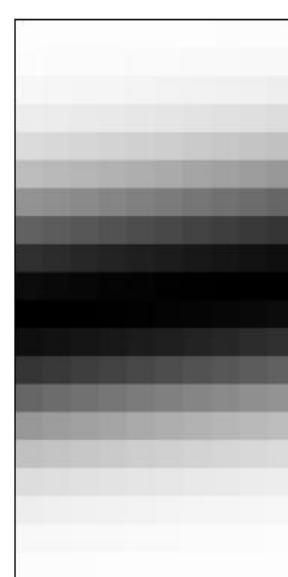
- For a data set of m dimensions, create m windows on the screen, one for each dimension
- The m dimension values of a record are mapped to m pixels at the corresponding positions in the windows
- The colors of the pixels reflect the corresponding values



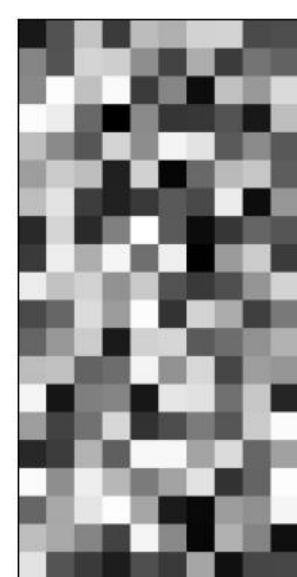
(a) Income



(b) Credit Limit



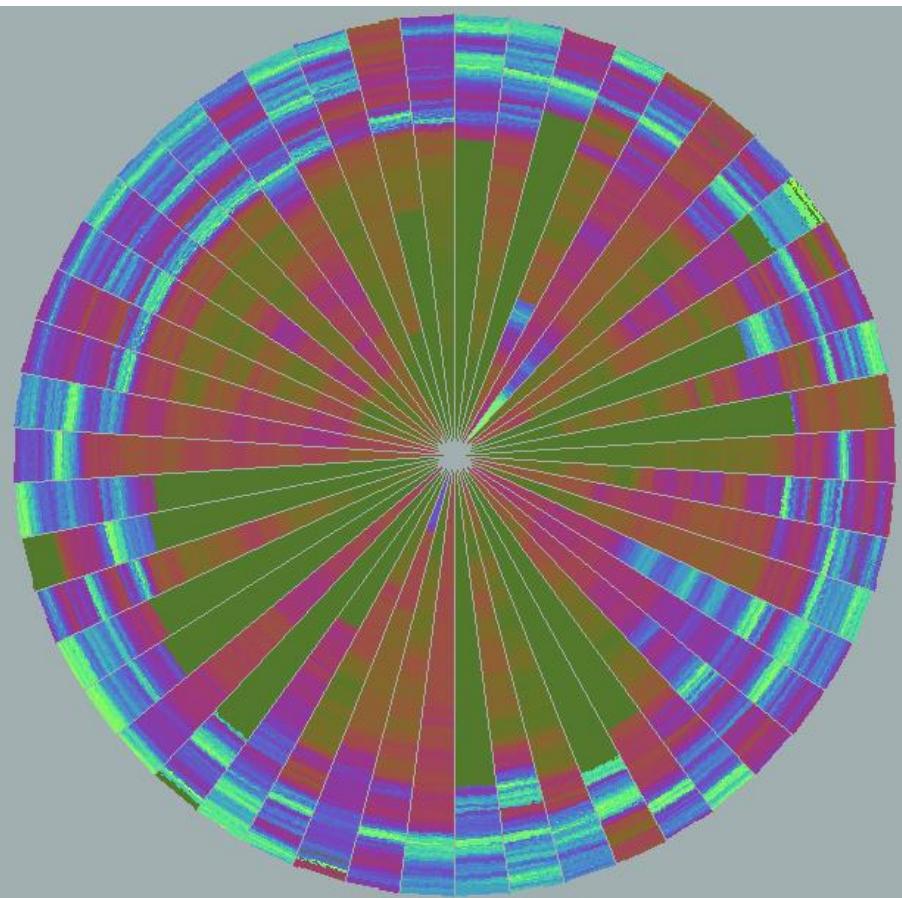
(c) transaction volume



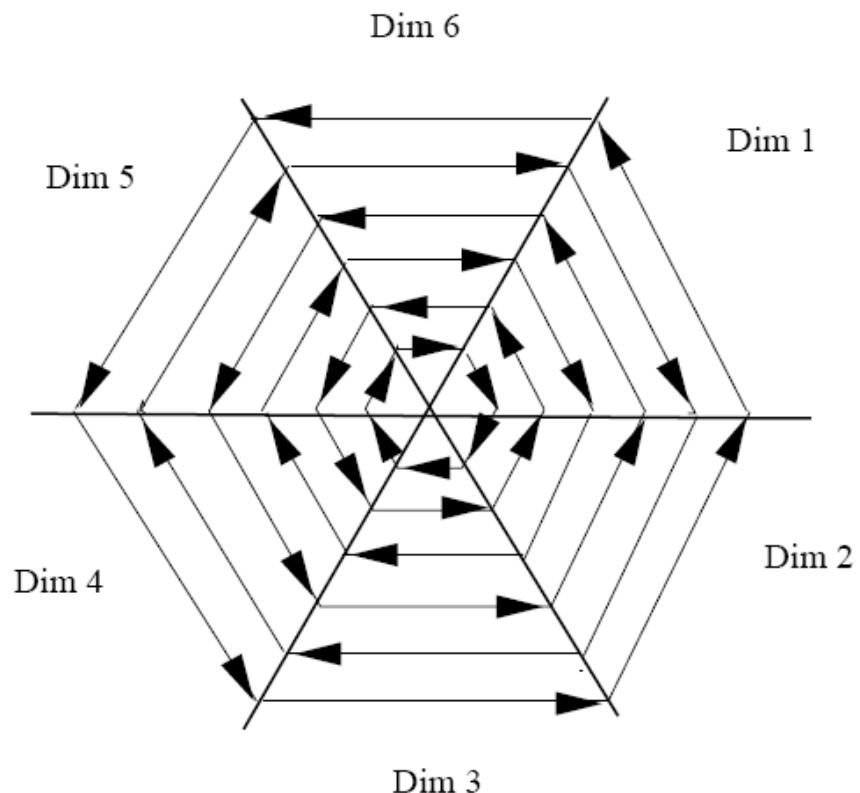
(d) age

Laying Out Pixels in Circle Segments

- To save space and show the connections among multiple dimensions, space filling is often done in a circle segment



Representing about 265,000 50-dimensional Data Items
with the 'Circle Segments' Technique



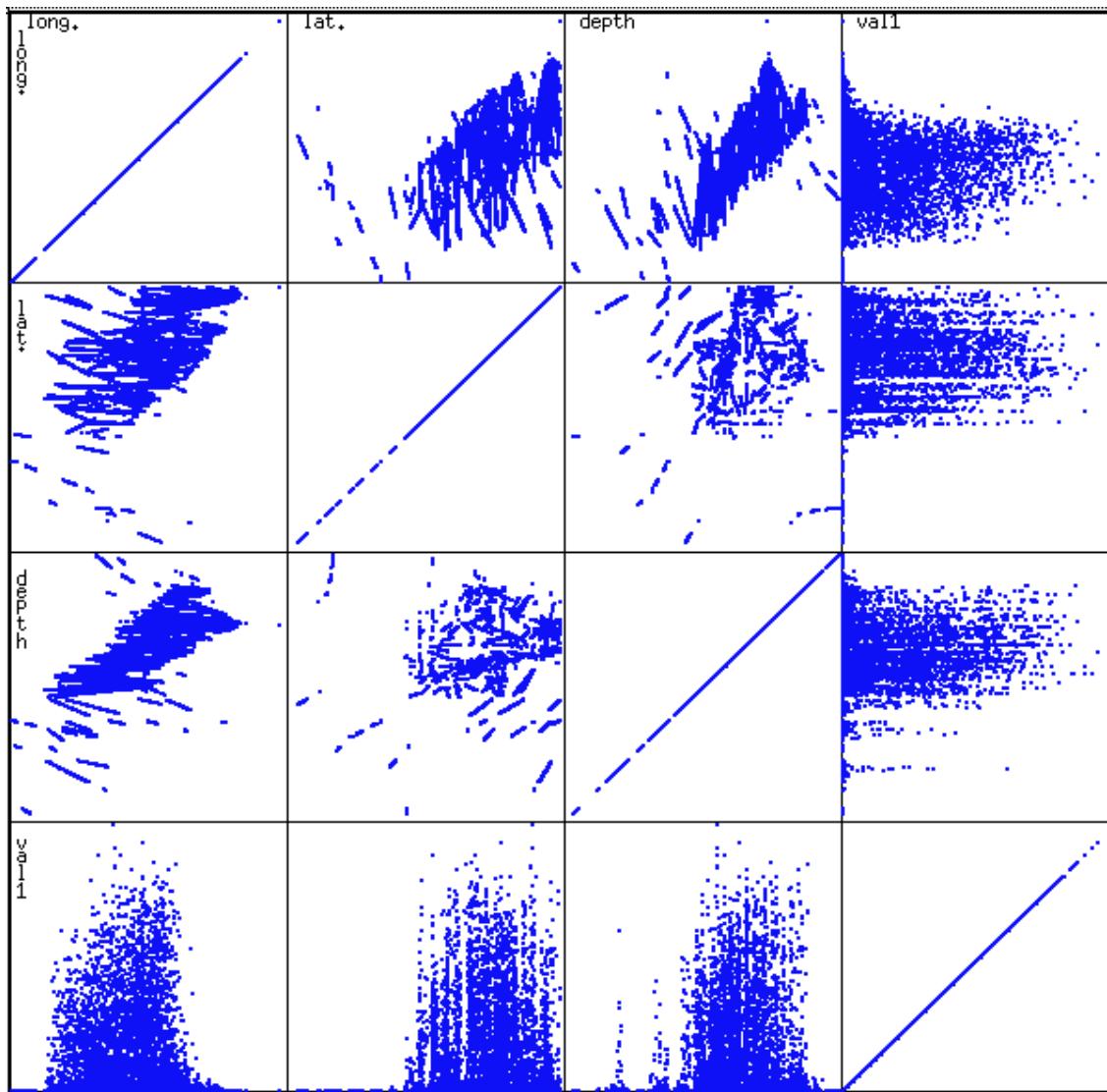
(b) Laying out pixels in circle segment

Geometric Projection Visualization Techniques

- Visualization of geometric transformations and projections of the data
- Methods
 - Direct visualization
 - Scatterplot and scatterplot matrices
 - Landscapes
 - Projection pursuit technique: Help users find meaningful projections of multidimensional data
 - Prosection views
 - Hyperslice
 - Parallel coordinates

Scatterplot Matrices

Used by permission of M. Ward, Worcester Polytechnic Institute

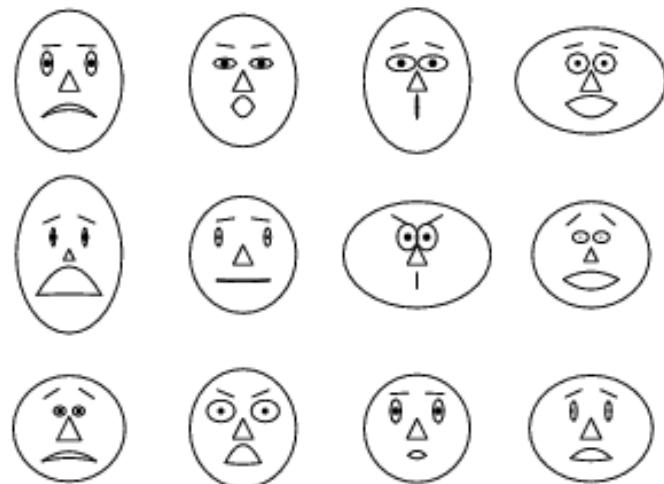


Matrix of scatterplots (x-y-diagrams) of the k-dim. data [total of $(k^2/2-k)$ scatterplots]

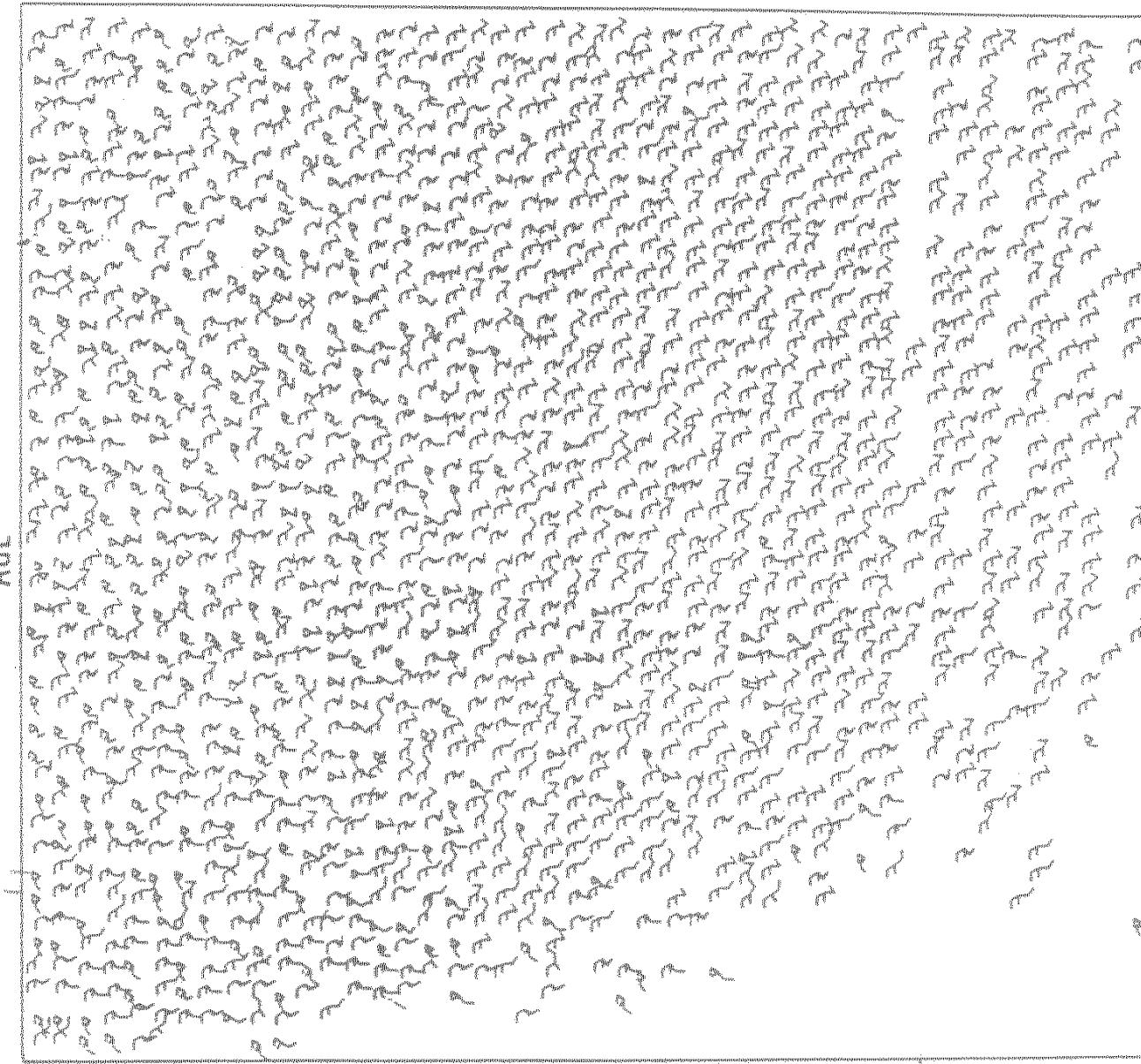
Chernoff Faces

- A way to display variables on a two-dimensional surface, e.g., let x be eyebrow slant, y be eye size, z be nose length, etc.
- The figure shows faces produced using 10 characteristics--head eccentricity, eye size, eye spacing, eye eccentricity, pupil size, eyebrow slant, nose size, mouth shape, mouth size, and mouth opening): Each assigned one of 10 possible values, generated using [*Mathematica*](#) (S. Dickson)

- REFERENCE: Gonick, L. and Smith, W. [*The Cartoon Guide to Statistics*](#). New York: Harper Perennial, p. 212, 1993
- Weisstein, Eric W. "Chernoff Face." From *MathWorld*--A Wolfram Web Resource.
mathworld.wolfram.com/ChernoffFace.html



Stick Figure



used by permission of G. Grinstein, University of Massachusetts at Lowell

A census data figure showing age, income, gender, education, etc.

A 5-piece stick figure (1 body and 4 limbs w. different angle/length)

Hierarchical Visualization Techniques

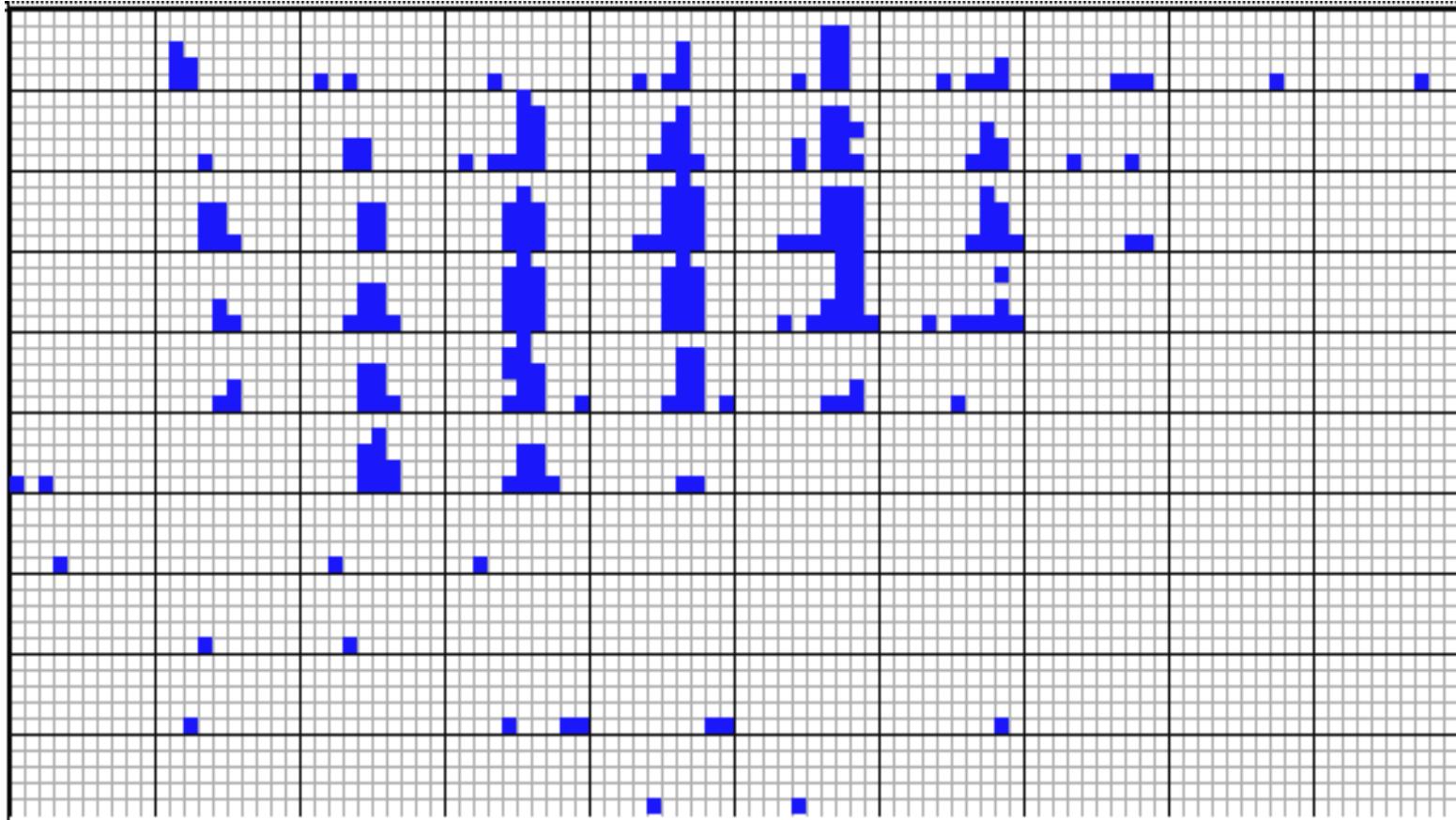
- Visualization of the data using a hierarchical partitioning into subspaces
- Methods
 - Dimensional Stacking
 - Worlds-within-Worlds
 - Tree-Map
 - Cone Trees
 - InfoCube

Dimensional Stacking

- Partitioning of the n-dimensional attribute space in 2-D subspaces, which are ‘stacked’ into each other
- Partitioning of the attribute value ranges into classes. The important attributes should be used on the outer levels.
- Adequate for data with ordinal attributes of low cardinality
- But, difficult to display more than nine dimensions
- Important to map dimensions appropriately

Dimensional Stacking

Used by permission of M. Ward, Worcester Polytechnic Institute



Visualization of oil mining data with longitude and latitude mapped to the outer x-, y-axes and ore grade and depth mapped to the inner x-, y-axes

Measuring data Similarity and Dissimilarity

- **Similarity**
 - Numerical measure of how alike two data objects are
 - Value is higher when objects are more alike
 - Often falls in the range [0,1]
- **Dissimilarity** (e.g., distance)
 - Numerical measure of how different two data objects are
 - Lower when objects are more alike
 - Minimum dissimilarity is often 0
 - Upper limit varies
- **Proximity** refers to a similarity or dissimilarity

Proximity Measure for Nominal Attributes

Method 1: Simple matching

- m : number of matches,
- p : total number of variables

$$d(i, j) = \frac{p - m}{p}$$

Example:

Objects	code
1	Code A
2	Code B
3	Code C
4	Code A

Proximity Measure for Nominal Attributes

Objects	code
1	Code A
2	Code B
3	Code C
4	Code A

$$\begin{matrix} & & 0 \\ d(2,1) & & 0 \\ d(3,1) & d(3,2) & 0 \\ d(4,1) & d(4,2) & d(4,3) & 0 \end{matrix}$$

$$d(i,j) = \frac{p-m}{p}$$



- $d(i,j)$ is 0 if objects I and j match
- $d(i,j)$ is 1 if objects I and j differ

$$\begin{matrix} 0 \\ 1 & 0 \\ 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \end{matrix}$$

Proximity Measure for Binary Attributes

- Distance measure for symmetric binary variables:

$$d(i, j) = \frac{r + s}{q + r + s + t}$$

- Distance measure for asymmetric binary variables:

$$d(i, j) = \frac{r + s}{q + r + s}$$

		Object <i>j</i>		
		1	0	sum
Object <i>i</i>	1	<i>q</i>	<i>r</i>	<i>q + r</i>
	0	<i>s</i>	<i>t</i>	<i>s + t</i>
	sum	<i>q + s</i>	<i>r + t</i>	<i>p</i>

q:11
r: 10
s: 01
t: 00

Dissimilarity between Binary Variables

- Example

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Gender is a symmetric attribute
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N be 0

$$d(jack, mary) = \frac{0 + 1}{2 + 0 + 1} = 0.33$$

$$d(jack, jim) = \frac{1 + 1}{1 + 1 + 1} = 0.67$$

$$d(jim, mary) = \frac{1 + 2}{1 + 1 + 2} = 0.75$$

Output: results for
Jack and mary are
more similar

Dissimilarity of Numeric data

- Distance measure are commonly used for computing the dissimilarity of objects described by numeric attribute
- These measures include the Euclidean, Manhattan and Minkowski distance

Dissimilarity of Numeric data

- Let i and j are two objects described by p numeric attributes

$$i = (x_{i1}, x_{i2}, \dots, x_{ip})$$

$$j = (x_{j1}, x_{j2}, \dots, x_{jp})$$

- The Euclidean distance between objects i and j is defined as

$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

Dissimilarity of Numeric data

- Manhattan distance

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

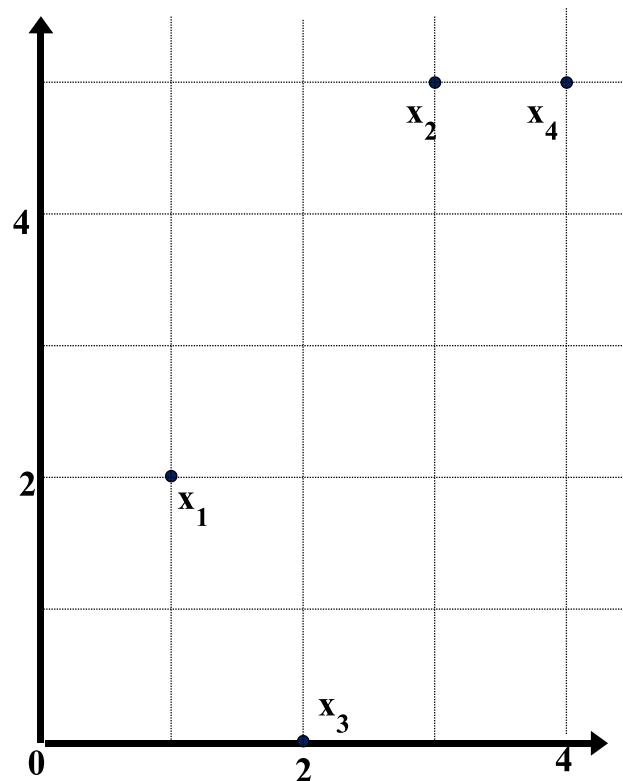
- Minkowski distance

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \dots + |x_{ip} - x_{jp}|^h}$$

Example: Data Matrix and Dissimilarity Matrix

Data Matrix

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

$$d(x1, x1) = (1-1) + (2-2) = 0$$

$$d(x2, x1) = (3-1) + (5-2) = 5$$

$$d(x3, x1) = (2-1) + (0-2) = 3$$

$$d(x4, x1) = (4-1) + (5-2) = 6$$

Cosine Similarity

- A **document** can be represented by thousands of attributes, each recording the *frequency* of a particular word (such as keywords) or phrase in the document.

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

- 0 value means that document do not share the word
- But this does not make them similar
- We need a measure that will focus on the words that the two documents do have in common.
- Cosine similarity is a measure of similarity that can be used to compare documents

Example: Cosine Similarity

- $\text{Cos}(x, y) = \text{sim}(x, y) = (x \bullet y) / \|x\| \|y\|$
→ where \bullet indicates vector dot product,

$$\rightarrow \|x\| : (x_1 * x_1 + x_2 * x_2 + \dots + x_p * x_p)^{0.5}$$

- Ex: Find the **similarity** between documents 1 and 2.

$$d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$$

$$d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$$

$$d_1 \bullet d_2 = 5*3+0*0+3*2+0*0+2*1+0*1+0*1+2*1+0*0+0*1 = 25$$

$$\|d_1\| = (5*5+0*0+3*3+0*0+2*2+0*0+0*0+2*2+0*0+0*0)^{0.5} = (42)^{0.5} = 6.481$$

$$\|d_2\| = (3*3+0*0+2*2+0*0+1*1+1*1+0*0+1*1+0*0+1*1)^{0.5} = (17)^{0.5} = 4.12$$

$$\cos(d_1, d_2) = 25 / (6.481 * 4.12)$$

$$\cos(d_1, d_2) = 0.94$$

Cosine Similarity

- Cosine value of zero means that two vectors are at 90 degree to each other and have no match
- The closer the cosine value to 1, the smaller is the angle and greater is the match between vectors

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
 - replace x_{if} by their rank $r_{if} \in \{1, \dots, M_f\}$
 - map the range of each variable onto $[0, 1]$ by replacing i -th object in the f -th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

- compute the dissimilarity using methods for interval-scaled variables

Attributes of Mixed Type

- A database may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, ordinal
- One may use a weighted formula to combine their effects

$$d(i, j) = \frac{\sum_{f=1}^p \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^p \delta_{ij}^{(f)}}$$

- f is binary or nominal:
 $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ otherwise
- f is numeric: use the normalized distance
- f is ordinal
 - Compute ranks r_{if} and
 - Treat z_{if} as interval-scaled

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

Chapter 2: Getting to Know Your Data

- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization
- Measuring Data Similarity and Dissimilarity
- Summary 

Chapter 2: Data Preprocessing

Why Data Preprocessing?

- Data in the real world is dirty
 - **incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
 - e.g., occupation=""
 - **noisy**: containing errors or outliers
 - e.g., Salary="-10"
 - **inconsistent**: containing discrepancies in codes or names
 - e.g., Age="30" Birthday="03/07/1980"
 - e.g., Was rating "1,2,3", now rating "A, B, C"
 - e.g., discrepancy between duplicate records

Why Is Data Dirty?

- Incomplete data may come from
 - “Not applicable” data value when collected
 - Different considerations between the time when the data was collected and when it is analyzed.
 - Human/hardware/software problems
- Noisy data (incorrect values) may come from
 - Faulty data collection instruments
 - Human or computer error at data entry
 - Errors in data transmission
- Inconsistent data may come from
 - Different data sources
 - Functional dependency violation (e.g., modify some linked data)
- Duplicate records also need data cleaning

Why Is Data Preprocessing Important?

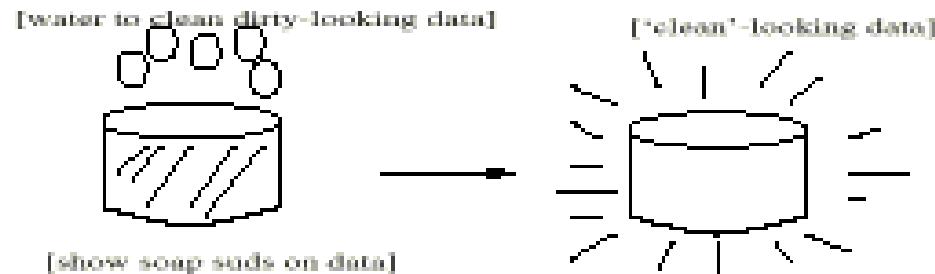
- No quality data, no quality mining results!
 - Quality decisions must be based on quality data
 - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
 - Data warehouse needs consistent integration of quality data
 - Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse

Major Tasks in Data Preprocessing

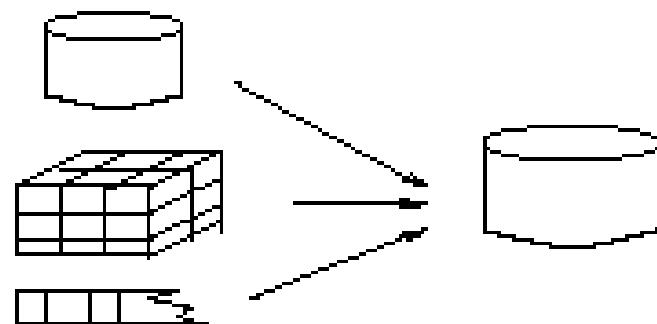
- Data cleaning
 - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
 - Integration of multiple databases, data cubes, or files
- Data transformation
 - Normalization and aggregation
- Data reduction
 - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization
 - Part of data reduction but with particular importance, especially for numerical data

Forms of Data Preprocessing

Data Cleaning



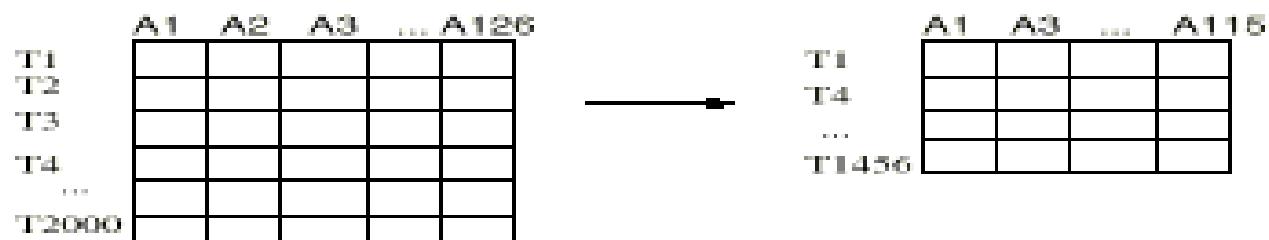
Data Integration



Data Transformation

-2, 32, 100, 59, 48 → -0.02, 0.32, 1.00, 0.59, 0.48

Data Reduction



Data Cleaning

- Importance
 - “Data cleaning is one of the three biggest problems in data warehousing”
- **Data Cleaning tasks**
 - Fill in missing values
 - Identify outliers and smooth out noisy data
 - Correct inconsistent data
 - Resolve redundancy caused by data integration

Missing Data

- Data is not always available
 - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
 - equipment malfunction
 - inconsistent with other recorded data and thus deleted
 - data not entered due to misunderstanding
 - certain data may not be considered important at the time of entry
 - not register history or changes of the data
- Missing data may need to be inferred.

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
 - a global constant : e.g., “unknown”
 - the attribute mean
 - the attribute mean for all samples belonging to the same class: smarter
 - the most probable value: inference-based such as **Bayesian formula or decision tree**

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to..
 - faulty data collection instruments
 - data entry problems
 - data transmission problems
 - technology limitation
 - inconsistency in naming convention

How to Handle Noisy Data?

- **Binning**

- first sort data and partition into (equal-frequency) bins
- then one can **smooth by bin means, smooth by bin median, smooth by bin boundaries**, etc.

- **Regression**

- smooth by fitting the data into regression functions

- **Clustering**

- detect and remove outliers

Binning method

- **Equal-depth** (frequency) partitioning
 - Divides the range into N intervals, each containing approximately same number of samples
 - Good data scaling
 - Managing categorical attributes can be tricky
- **Equal-width** (distance) partitioning
 - Divides the range into N intervals of equal size
 - if A and B are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
 - The most straightforward, but outliers may dominate presentation
 - Skewed data is not handled well

Binning Methods for Data Smoothing (Example)

Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34

* Partition into equal-frequency (equi-depth) bins:

- Bin 1: 4, 8, 9, 15
- Bin 2: 21, 21, 24, 25
- Bin 3: 26, 28, 29, 34

1) Smoothing by bin means:

- Bin 1: 9, 9, 9, 9
- Bin 2: 23, 23, 23, 23
- Bin 3: 29, 29, 29, 29

2) Smoothing by bin boundaries:

- Bin 1: **4**, 4, 4, **15**
- Bin 2: **21**, 21, 25, **25**
- Bin 3: **26**, 26, 26, **34**

Q) Suppose a group of 12 *sales price* records has been sorted as follows:

5; 10; 11; 13; 15; 35; 50; 55; 72; 92; 204; 215;

Partition them into **three bins** by each of the following methods.

- (a) equal-frequency partitioning
- (b) equal-width partitioning
- (c) clustering

(a) equal-frequency partitioning

bin 1 -- 5,10,11,13

bin 2 -- 15,35,50,55

bin 3 -- 72,92,204,215

(b) equal-width partitioning

The width of each interval is $(215 - 5)/3 = 70$.

bin 1 -- 5,10,11,13,15,35,50,55,72 (5 to 75)

bin 2 -- 92 (76 to 146)

bin 3 -- 204,215 (147 to 217)

(c) clustering

We will use a simple clustering technique:

Partition the data along the 2 biggest gaps in the data.

bin 1 5,10,11,13,15

bin 2 35,50,55,72,92

bin 3 204,215

(ex. 5; 10; 11; 13; 15; 35; 50; 55; 72; 92; 204; 215;)

Data Integration

- Data integration:
 - Combines data from multiple sources into a coherent store
- Schema integration: e.g., A.cust-id \equiv B.cust-#
 - Integrate metadata from different sources
- Entity identification problem:
 - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- Detecting and resolving data value conflicts
 - For the same real world entity, attribute values from different sources are different
 - Possible reasons: different representations, different scales, e.g., metric vs. British units

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
 - *Object identification:* The same attribute or object may have different names in different databases
 - *Derivable data:* One attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by correlation analysis
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Redundancy and Correlation Analysis

- Some redundancy can be detected by correlation analysis.
- Such analysis can measure how strongly one attributes implies the other based on the available data
 - For nominal data, χ^2 (chi-square) test
 - For numeric data correlation coefficient and covariance

Correlation Analysis (Categorical Data)

- χ^2 (chi-square) test

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- $Expected = \frac{(count\ A)*(count\ B)}{n}$
- The larger the χ^2 value, the more likely the variables are related

Chi-Square Calculation: An Example

	Play chess	Not play chess	Sum (row): A
Like science fiction	250 (exp: 90)	200 (exp:360)	450
Not like science fiction	50 (exp:210)	1000(exp:840)	1050
Sum(col.): B	300	1200	1500 (n)

$$expected = \frac{(count\ A)*(count\ B)}{n} = \frac{450*300}{1500} = 90$$

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- It shows that like_science_fiction and play_chess are correlated in the group

- We can get χ^2 by:

$$\begin{aligned}\chi^2 &= \frac{(250 - 90)^2}{90} + \frac{(50 - 210)^2}{210} + \frac{(200 - 360)^2}{360} + \frac{(1000 - 840)^2}{840} \\ &= 284.44 + 121.90 + 71.11 + 30.48 = 507.93.\end{aligned}$$

- For this 2×2 table, the degrees of freedom are $(2-1)(2-1) = 1$.
- For 1 degree of freedom, the χ^2 value needed to reject the hypothesis at the 0.001 significance level is 10.828 (taken from the table of upper percentage points of the χ^2 distribution, typically available from any textbook on statistics).

$507.93 > 10.82$

Calculated value is more than tabulated value of χ^2

So,

Null hypothesis: play chess and preferred reading are independent
(not related)

is rejected and conclude that the two attributes are strongly correlated for the given group of people

Correlation Analysis (Numeric Data)

- Correlation coefficient (also called Pearson's product moment coefficient)

$$r_{A,B} = \frac{Cov(A, B)}{\sigma_A \sigma_B}$$

- If $r_{A,B} > 0$,
A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent;
- $r_{A,B} < 0$: negatively correlated

Covariance (Numeric Data)

$$\text{Cov}(A, B) = E(A \cdot B) - \bar{A}\bar{B}$$

where \bar{A} and \bar{B} are the respective mean or **expected values** of A and B

Positive covariance: If $\text{Cov}_{A,B} > 0$, then A and B both tend to be larger than their expected values

Negative covariance: If $\text{Cov}_{A,B} < 0$ then if A is larger than its expected value, B is likely to be smaller than its expected value

Independence: $\text{Cov}_{A,B} = 0$ but the converse is not true:

Some pairs of random variables may have a covariance of 0 but are not independent. Only under some additional assumptions (e.g., the data follow multivariate normal distributions) does a covariance of 0 imply independence

Co-Variance: An Example

- It can be simplified in computation as

$$Cov(A, B) = E(A \cdot B) - \bar{A}\bar{B}$$

Example:

- Suppose two stocks A and B have the following values in one week: (2, 5), (3, 8), (5, 10), (4, 11), (6, 14).
- **Question: If the stocks are affected by the same industry trends, will their prices rise or fall together?**

• $\bar{A} = (2 + 3 + 5 + 4 + 6) / 5 = 20/5 = 4$

• $\bar{B} = (5 + 8 + 10 + 11 + 14) / 5 = 48/5 = 9.6$

• $Cov(A, B) = (2 \times 5 + 3 \times 8 + 5 \times 10 + 4 \times 11 + 6 \times 14) / 5 - 4 \times 9.6 = 4$

- Thus, A and B rise together since $Cov(A, B) > 0$.

Days	Stock A	Stock B
Monday	2	5
Tuesday	3	8
Wednesday	5	10
Thursday	4	11
Friday	6	14

Data Transformation

- A function that maps the entire set of values of a given attribute to a new set of replacement values so that each old value can be identified with one of the new values
- Methods
 - **Smoothing:** Remove noise from data
 - Techniques include binning, regression, clustering
 - **Attribute/feature construction**
 - New attributes constructed from the given ones
 - **Aggregation:** Summarization, data cube construction
 - Eg. Daily sales data aggregated to monthly and annual income

Data Transformation

- **Normalization:** attribute data are scaled to fall within a smaller range such as (-1.0 to 1.0) or (0.0 to 1.0)
 - min-max normalization
 - z-score normalization
 - normalization by decimal scaling
- **Discretization:** raw values of numeric attributes (e.g. age) are replaced by interval labels (0-10, 11-20) or conceptual labels(youth, adult, senior) resulting in Concept hierarchy for the numeric attributes
- **Concept hierarchy generation for nominal data:**
- Attribute such as street can be generalized to higher level concept, like city or country

Normalization

- **Min-max normalization:** it maps a value v of attribute A to new value v' in the range $[new_min_A, new_max_A]$ by computing,

$$v' = \frac{v - min_A}{max_A - min_A} (new_max_A - new_min_A) + new_min_A$$

- Ex. Let min and max value for the attribute income are \$12,000 and \$98,000 resp. Now map income to the range [0.0, 1.0]. Then the value \$73,600 for income is transformed as,

$$\frac{73,600 - 12,000}{98,000 - 12,000} (1.0 - 0) + 0 = 0.716$$

Normalization

- **Z-score normalization:** The values for attribute A are normalized based on mean and (μ) standard deviation(σ) of attribute A .
Formula is given by,

$$v' = \frac{v - \mu_A}{\sigma_A}$$

- Ex. Let $\mu = 54,000$, $\sigma = 16,000$ for attribute income. With z-score normalization, a value \$73600 for income is transformed to ,
$$\frac{73,600 - 54,000}{16,000} = 1.225$$

Normalization

- **Normalization by decimal scaling:** It transforms the value by moving the decimal point of value of attribute A. The number of decimal points moved depends on the maximum absolute value of A.

- formula is given by, $v' = \frac{v}{10^j}$ Where j is the smallest integer such that $\text{Max}(|v'|) < 1$

- Eg. Let the range for attribute A be -986 to 927. To normalize by decimal scaling, divide each value by 1000(i.e. j=3).
Therefore -986 is normalized to -0.986 and 917 is normalized to 0.917.

Example

Use the two methods below to *normalize* the following group of data:

200; 300; 400; 600; 1000

- (a) min-max normalization by setting $\min = 0$ and $\max = 1$
- (b) z-score normalization

<i>age</i>	23	23	27	27	39	41	47	49	50
<i>z-age</i>	-1.83	-1.83	-1.51	-1.51	-0.58	-0.42	0.04	0.20	0.28
<i>%fat</i>	9.5	26.5	7.8	17.8	31.4	25.9	27.4	27.2	31.2
<i>z-%fat</i>	-2.14	-0.25	-2.33	-1.22	0.29	-0.32	-0.15	-0.18	0.27
<i>age</i>	52	54	54	56	57	58	58	60	61
<i>z-age</i>	0.43	0.59	0.59	0.74	0.82	0.90	0.90	1.06	1.13
<i>%fat</i>	34.6	42.5	28.8	33.4	30.2	34.1	32.9	41.2	35.7
<i>z-%fat</i>	0.65	1.53	0.0	0.51	0.16	0.59	0.46	1.38	0.77

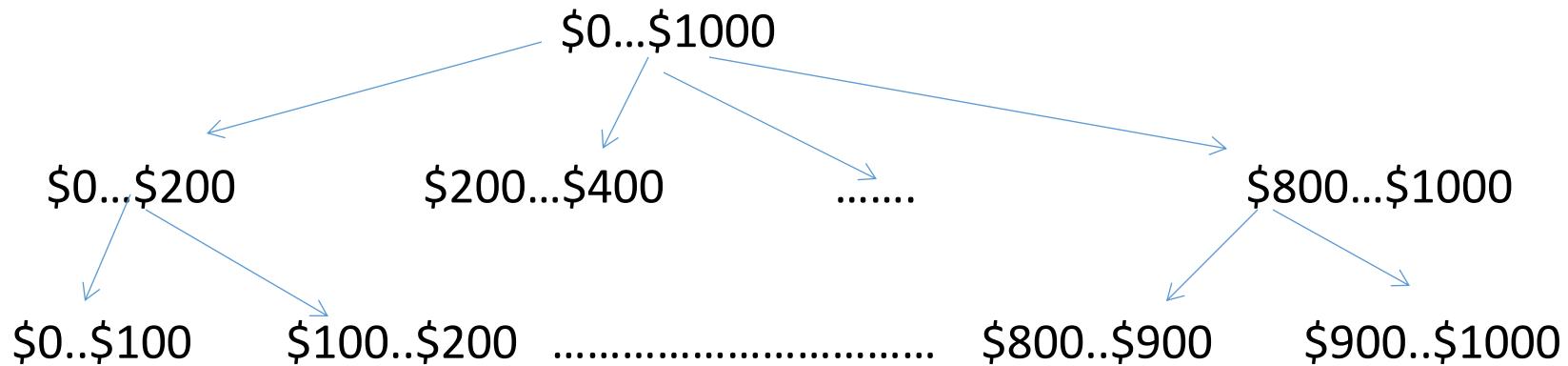
Using the data for *age* given in Exercise 2.4, answer the following:

- (a) Use min-max normalization to transform the value 35 for *age* onto the range [0:0; 1:0]
- (b) Use z-score normalization to transform the value 35 for *age*, where the standard deviation of *age* is 12.94 years.
- (c) Use normalization by decimal scaling to transform the value 35 for *age*.

Concept Hierarchy Generation

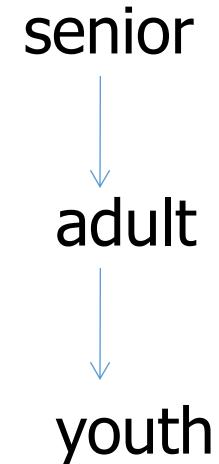
- **Concept hierarchy** organizes concepts (i.e., attribute values) hierarchically and is usually associated with each dimension in a data warehouse
- Concept hierarchies facilitate drilling and rolling in data warehouses to view data in multiple granularity
- Concept hierarchy formation: Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for *age*) by higher level concepts (such as *youth*, *adult*, or *senior*)
- Concept hierarchies can be explicitly specified by domain experts and/or data warehouse designers

Concept hierarchy for the numeric attributes (Discretization)



Concept hierarchy for attribute price
(interval labels)

Concept hierarchy for attribute Age
(conceptual labels)

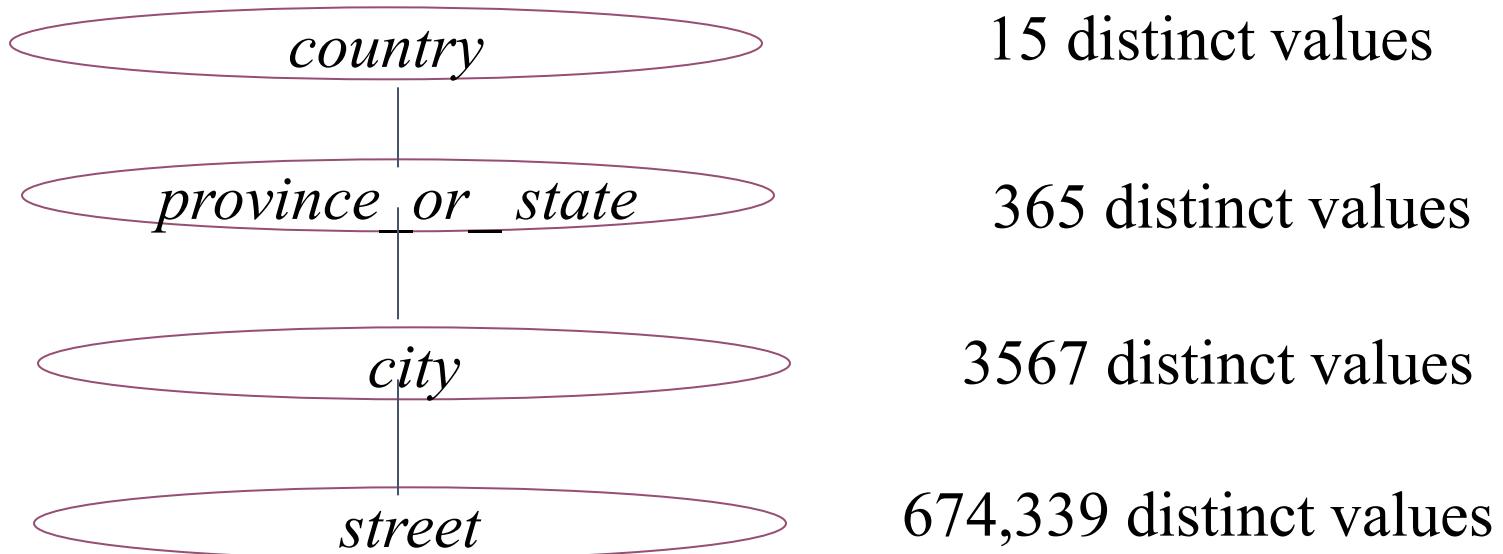


Concept Hierarchy Generation for Nominal Data

- 1. Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts.**
 - *street < city < state < country*
- 2. Specification of a hierarchy for a set of values by explicit data grouping**
 - {Punjab, Haryana, Delhi} < North_India
 - {Karnataka, Tamilnadu, kerala} < South_India
- 3. Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values**

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
- The attribute with the most distinct values is placed at the lowest level of the hierarchy



Data Discretization Methods

- Typical methods: All the methods can be applied recursively
 - Binning
 - Top-down split, unsupervised
 - Histogram analysis
 - Top-down split, unsupervised
 - Clustering analysis (unsupervised, top-down split or bottom-up merge)
 - Decision-tree analysis (supervised, top-down split)
 - Correlation (e.g., χ^2) analysis (unsupervised, bottom-up merge)

Data Reduction

- Why data reduction?
 - A database/data warehouse may store terabytes of data
 - Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
 - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results

Data reduction strategies

- **Data cube aggregation:**
- Eg. Total sales per year instead of per quarter
- **Dimensionality reduction:** original data is transform into smaller space. Encoding methods are used.
- Eg. Wavelet transform and principal component analysis
- **Attribute subset selection:** remove unimportant(irrelevant, redundant, weakly relevant) attributes
- Eg. For new CD purchase, customers phone number is irrelevant

Data reduction strategies

- Numerosity reduction : replace original data by alternatives smaller form of data representation
- Parametric method: model is used to estimate the data
 - Eg. Regression, log-linear models
- Non Parametric method
 - Eg. Histograms, clustering, sampling and Data cube aggregation

Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
 - The aggregated data for an **individual entity of interest**
 - E.g., a customer in a phone calling data warehouse
- Multiple levels of aggregation in data cubes
 - Further reduce the size of data to deal with
- Reference appropriate levels
 - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

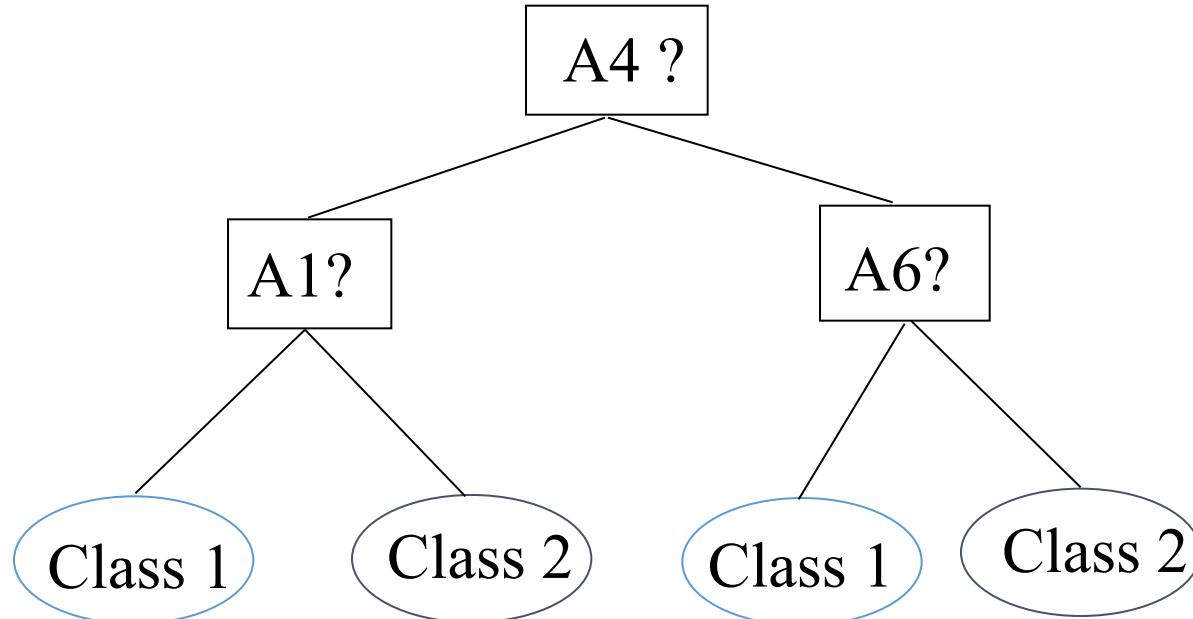
Attribute Subset Selection

- Feature selection (i.e., attribute subset selection):
 - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
 - reduce # of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
 - Step-wise forward selection
 - Step-wise backward elimination
 - Combining forward selection and backward elimination
 - Decision-tree induction

Example of Decision Tree Induction

Initial attribute set:

$\{A_1, A_2, A_3, A_4, A_5, A_6\}$



-----> Reduced attribute set: $\{A_1, A_4, A_6\}$

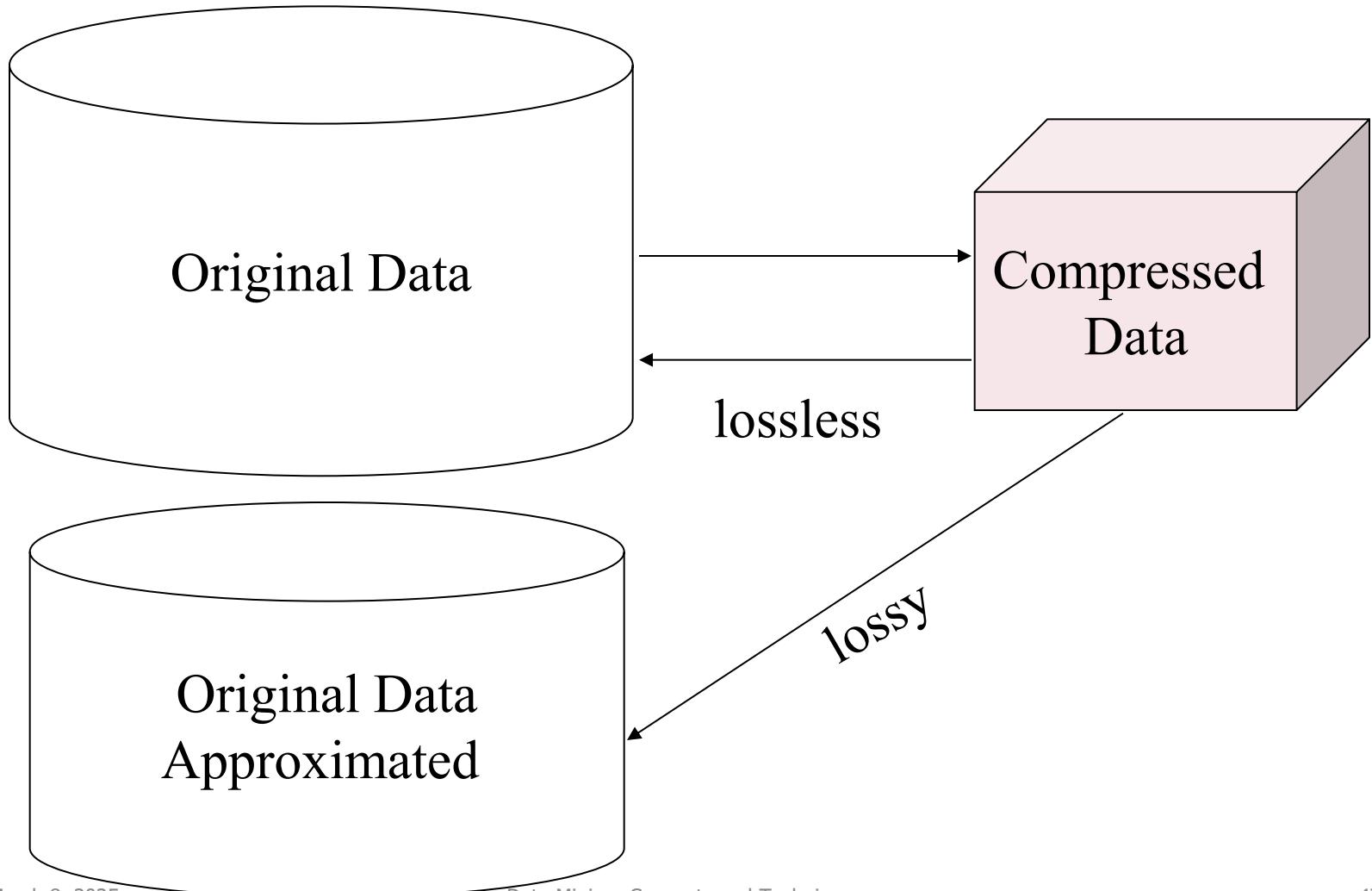
Heuristic Feature Selection Methods

- There are 2^d possible sub-features of d features
- Several heuristic feature selection methods:
 - Best single features under the feature independence assumption: choose by significance tests
 - Best step-wise feature selection:
 - The best single-feature is picked first
 - Then next best feature condition to the first, ...
 - Step-wise feature elimination:
 - Repeatedly eliminate the worst feature
 - Best combined feature selection and elimination
 - Optimal branch and bound:
 - Use feature elimination and backtracking

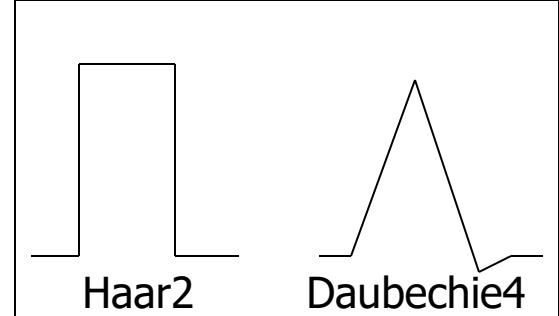
Data Compression

- String compression
 - There are extensive theories and well-tuned algorithms
 - Typically lossless
 - But only limited manipulation is possible without expansion
- Audio/video compression
 - Typically lossy compression, with progressive refinement
 - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
 - Typically short and vary slowly with time

Data Compression

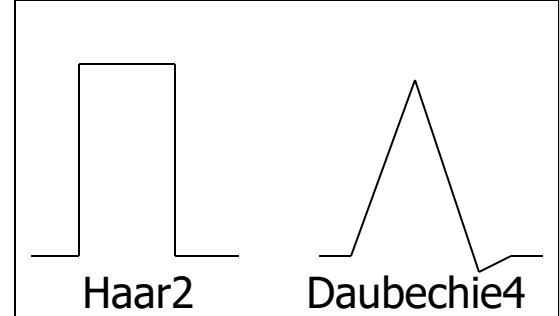


Dimensionality Reduction: Wavelet Transformation



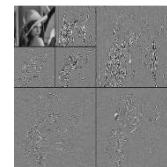
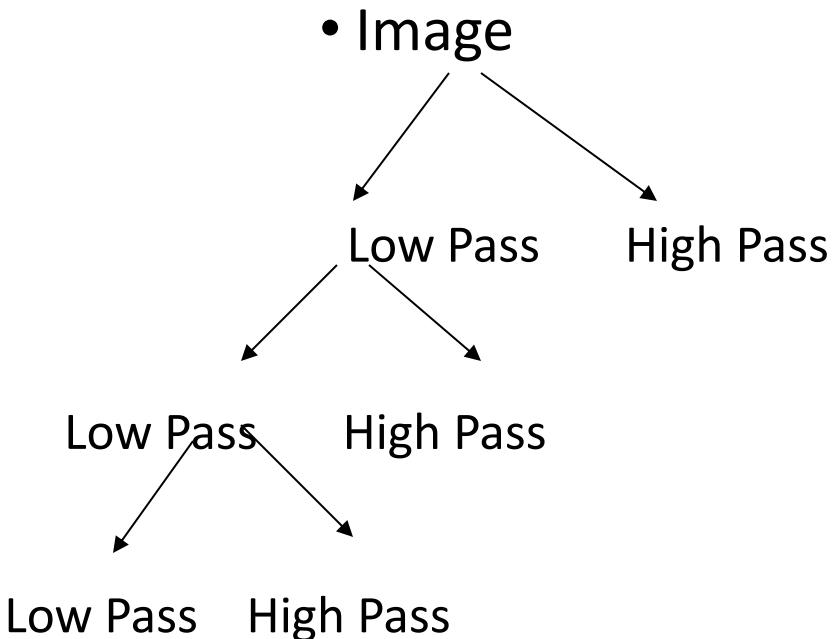
- Data encoding techniques or transformation are applied on the original data to obtain a reduced or compressed representation of the data.
- Reconstructed data can be lossy or lossless
- Lossy dimensionality reduction methods:
 - Wavelet transforms(DWT)
 - Haar transform
 - Principle component analysis(PCA)
 - K-L method

Dimensionality Reduction: Wavelet Transformation



- Discrete wavelet transform (DWT): linear signal processing, multi-resolutational analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
 - Length, L , must be an integer power of 2 (padding with 0's, when necessary)
 - Each transform has 2 functions: smoothing, difference
 - Applies to pairs of data, resulting in two set of data of length $L/2$
 - Applies two functions recursively, until reaches the desired length

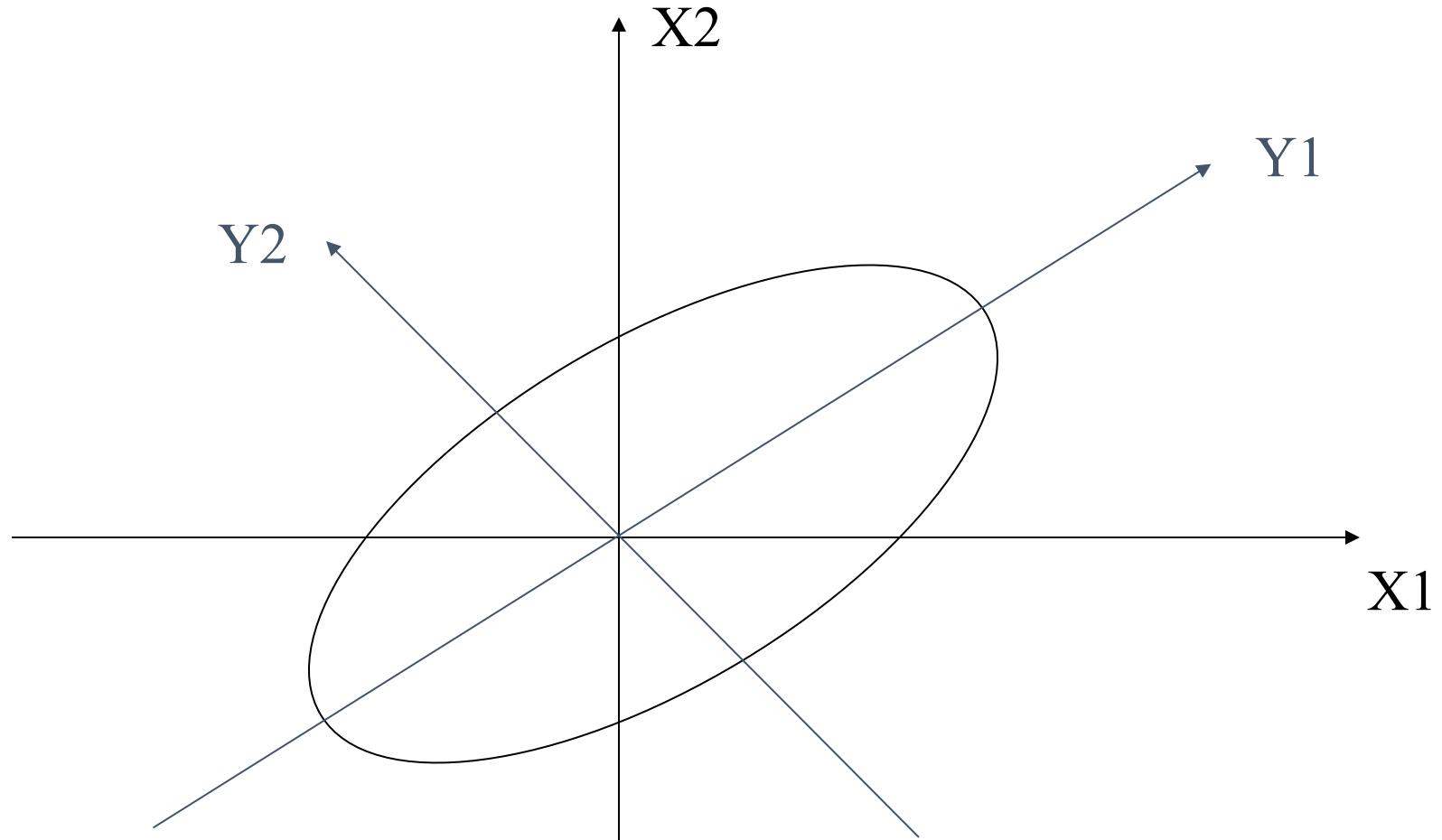
DWT for Image Compression



Dimensionality Reduction: Principal Component Analysis (PCA)

- Given N data vectors from n -dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
- Steps
 - Normalize input data: Each attribute falls within the same range
 - Compute k orthonormal (unit) vectors, i.e., *principal components*
 - Each input data (vector) is a linear combination of the k principal component vectors
 - The principal components are sorted in order of decreasing “significance” or strength
 - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance. (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data)
- Works for numeric data only
- Used when the number of dimensions is large

Principal Component Analysis



Numerosity Reduction

- Reduce data volume by choosing alternative, smaller forms of data representation
- Parametric methods
 - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
 - Example: Regression and Log-linear models—obtain value at a point in m-D space as the product on appropriate marginal subspaces
- Non-parametric methods
 - Do not assume models
 - Major families: histograms, clustering, sampling

Data Reduction Method (1): Regression and Log-Linear Models

- **Linear regression:** Data are modeled to fit a **straight line**

$$y = wx + b$$

Where, y and x → numerical database attributes

w and b → regression coefficient

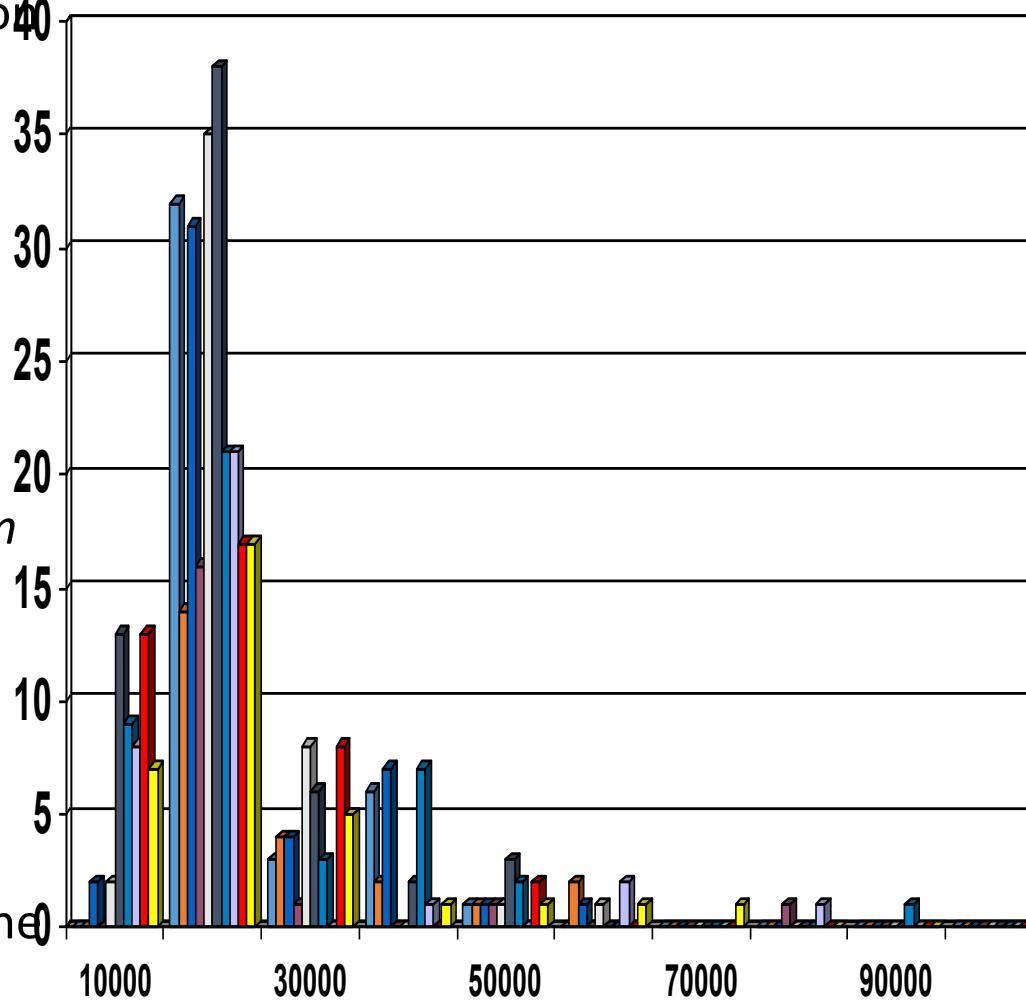
- **Multiple regression:** extension of linear regression method where y is modeled as a linear function of **two or more predictor variable**
- **Log-linear model:** estimate the **probability of each point** in a multidimensional space for a set of discretized attributes, based on the smaller subset of dimensional combinations

Regress Analysis and Log-Linear Models

- Linear regression: $Y = wX + b$
 - Two regression coefficients, w and b , specify the line and are to be estimated by using the data at hand
 - Using the least squares criterion to the known values of $Y_1, Y_2, \dots, X_1, X_2, \dots$
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$.
 - Many nonlinear functions can be transformed into the above
- Log-linear models:
 - The multi-way table of joint probabilities is approximated by a product of lower-order tables
 - Probability: $p(a, b, c, d) = \alpha ab \beta ac \gamma ad \delta bc d$

Data Reduction Method (2): Histograms

- Histogram for an attribute A → partitions data distribution of A into disjoint subsets or buckets.
- Partitioning rules:
 - Equal-width: equal bucket range
 - Equal-frequency (or equal-depth)
 - V-optimal: with the least *histogram variance* (weighted sum of the original values that each bucket represents)
 - MaxDiff: set bucket boundary between each pair for pairs have the $\beta - 1$ largest differences



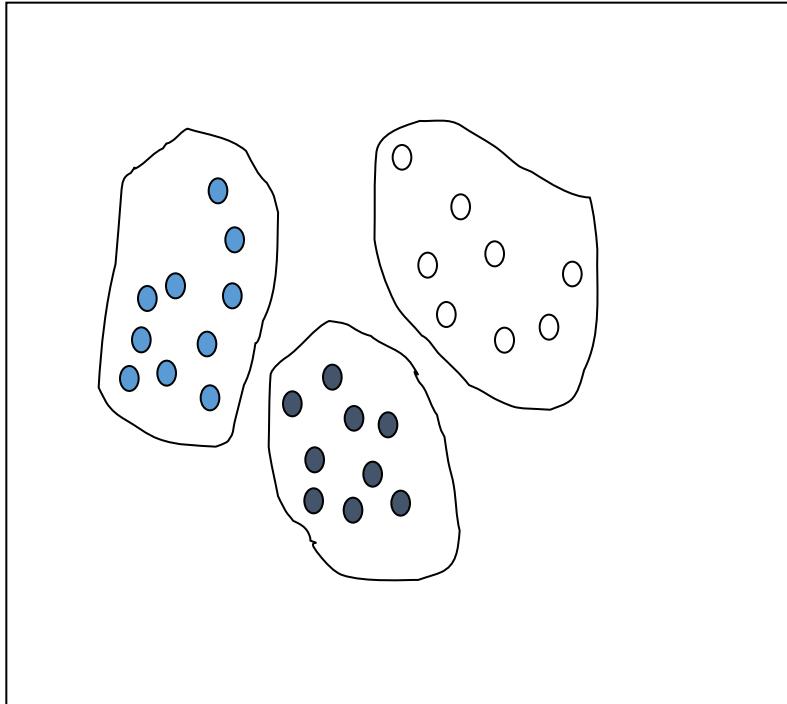
- List of prices of commonly sold items at AllElectronics(rounded to nearest dollar)
- 1,1,5,5,5,5,8,8,10,10,10,10,10,12,14,14,14,14,15,15,15,15,15

Data Reduction Method (3): Clustering

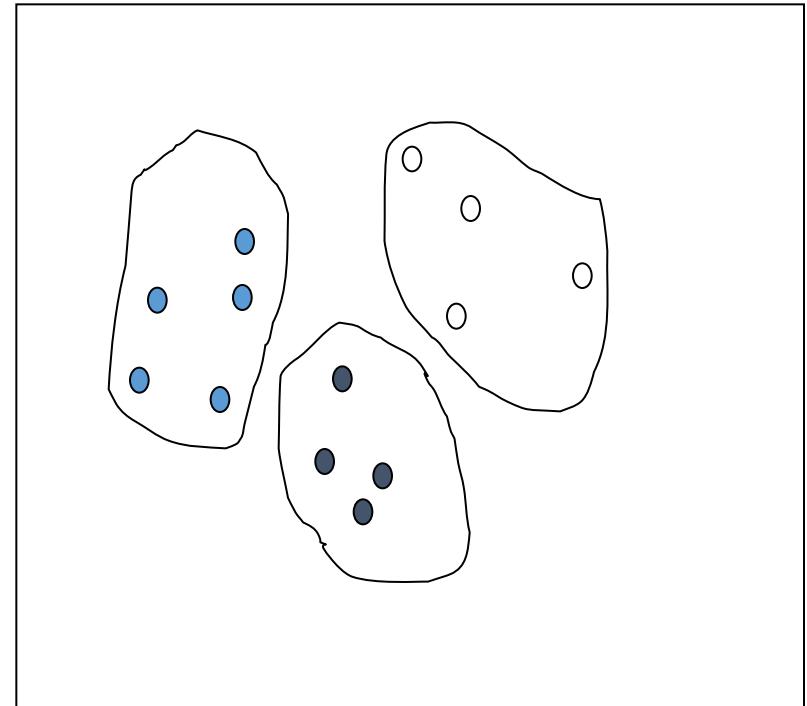
- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms
- Cluster analysis will be studied in depth in Chapter 7

Sampling: Cluster or Stratified Sampling

Raw Data



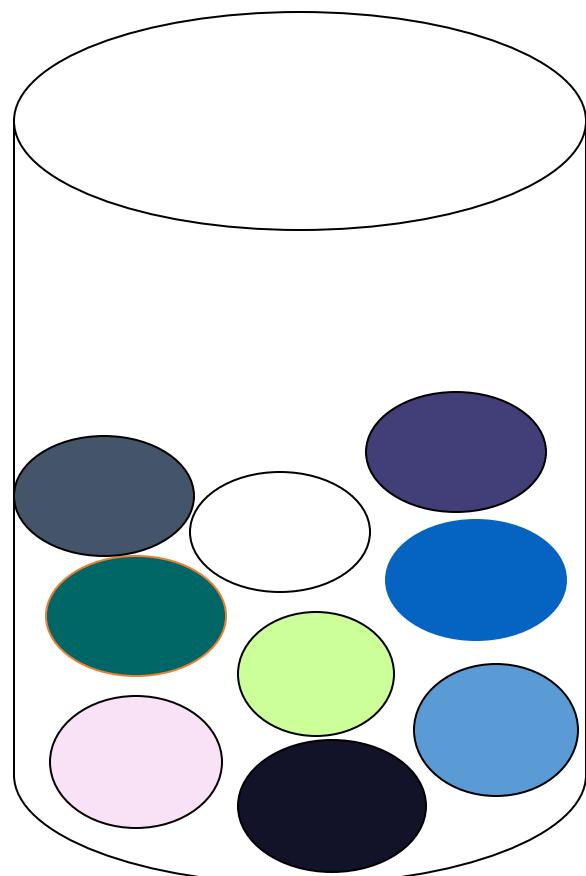
Cluster/Stratified Sample



Data Reduction Method (4): Sampling

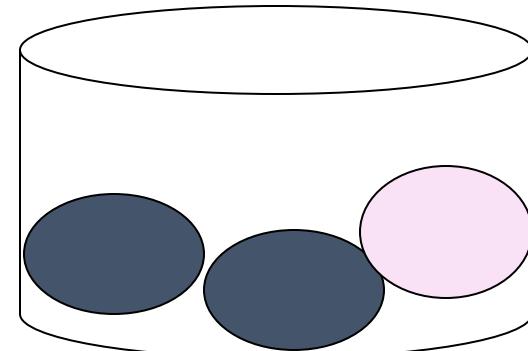
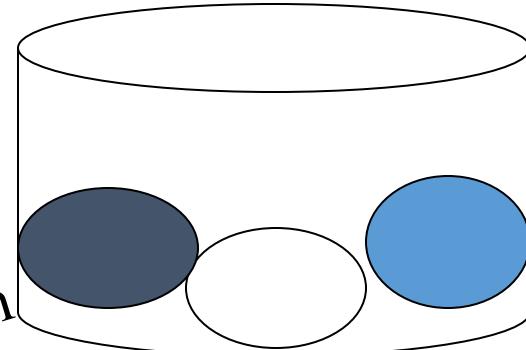
- Sampling: obtaining a small sample s to represent the whole data set N
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a **representative** subset of the data
 - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
 - Stratified sampling:
 - Approximate the percentage of each class (or subpopulation of interest) in the overall database
 - Used in conjunction with skewed data
- Note: Sampling may not reduce database I/Os (page at a time)

Sampling: with or without Replacement



SRSWOR
(simple random
sample without
replacement)

SRSWR



Data Mining: Concepts and Techniques

— Chapter 2 —

Jiawei Han

Department of Computer Science
University of Illinois at Urbana-Champaign

www.cs.uiuc.edu/~hanj

©2006 Jiawei Han and Micheline Kamber, All rights reserved



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Why Data Preprocessing?

- Data in the real world is dirty
 - **incomplete**: lacking attribute values, lacking certain attributes of interest, or containing only aggregate data
 - e.g., occupation=" "
 - **noisy**: containing errors or outliers
 - e.g., Salary="-10"
 - **inconsistent**: containing discrepancies in codes or names
 - e.g., Age="42" Birthday="03/07/1997"
 - e.g., Was rating "1,2,3", now rating "A, B, C"
 - e.g., discrepancy between duplicate records

Why Is Data Dirty?

- Incomplete data may come from
 - “Not applicable” data value when collected
 - Different considerations between the time when the data was collected and when it is analyzed.
 - Human/hardware/software problems
- Noisy data (incorrect values) may come from
 - Faulty data collection instruments
 - Human or computer error at data entry
 - Errors in data transmission
- Inconsistent data may come from
 - Different data sources
 - Functional dependency violation (e.g., modify some linked data)
- Duplicate records also need data cleaning

Why Is Data Preprocessing Important?

- No quality data, no quality mining results!
 - Quality decisions must be based on quality data
 - e.g., duplicate or missing data may cause incorrect or even misleading statistics.
 - Data warehouse needs consistent integration of quality data
- Data extraction, cleaning, and transformation comprises the majority of the work of building a data warehouse

Multi-Dimensional Measure of Data Quality

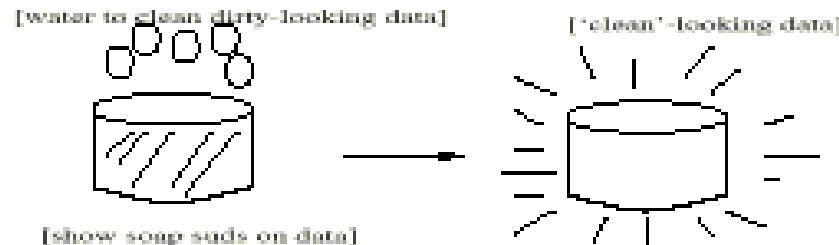
- A well-accepted multidimensional view:
 - Accuracy
 - Completeness
 - Consistency
 - Timeliness
 - Believability
 - Value added
 - Interpretability
 - Accessibility
- Broad categories:
 - Intrinsic, contextual, representational, and accessibility

Major Tasks in Data Preprocessing

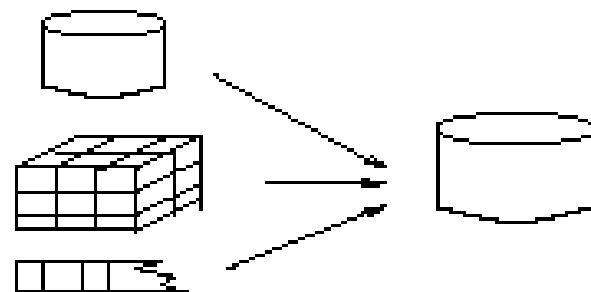
- Data cleaning
 - Fill in missing values, smooth noisy data, identify or remove outliers, and resolve inconsistencies
- Data integration
 - Integration of multiple databases, data cubes, or files
- Data transformation
 - Normalization and aggregation
- Data reduction
 - Obtains reduced representation in volume but produces the same or similar analytical results
- Data discretization
 - Part of data reduction but with particular importance, especially for numerical data

Forms of Data Preprocessing

Data Cleaning



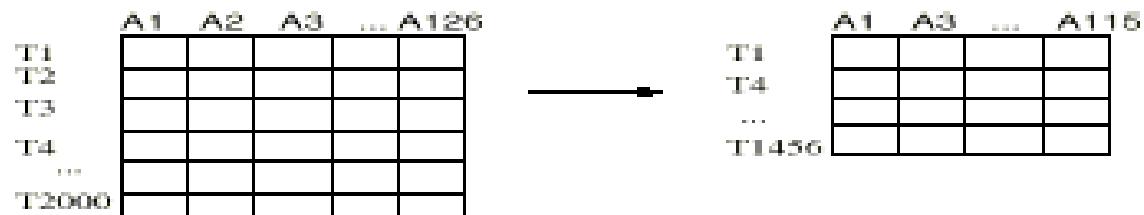
Data Integration



Data Transformation

-2, 32, 100, 59, 48 → -0.02, 0.32, 1.00, 0.59, 0.48

Data Reduction



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Mining Data Descriptive Characteristics

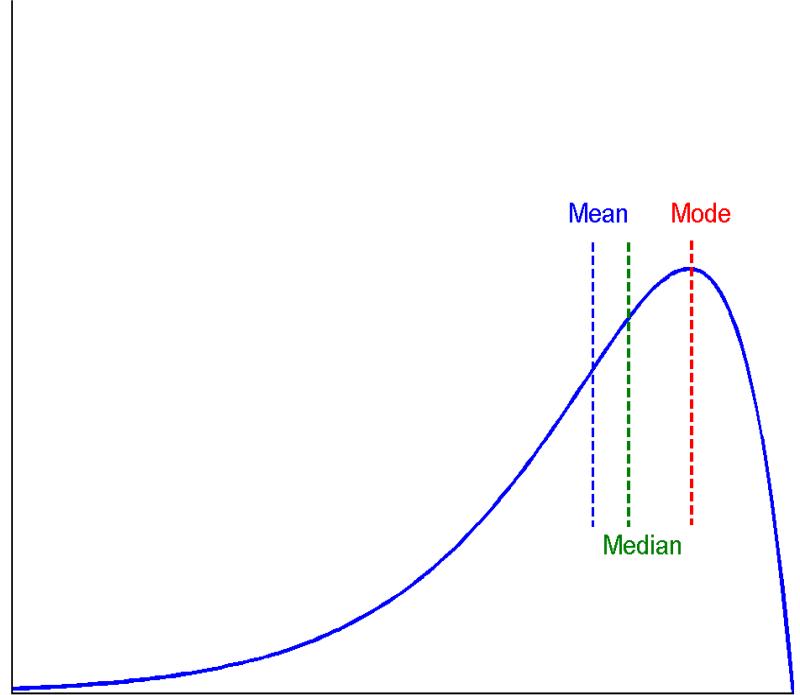
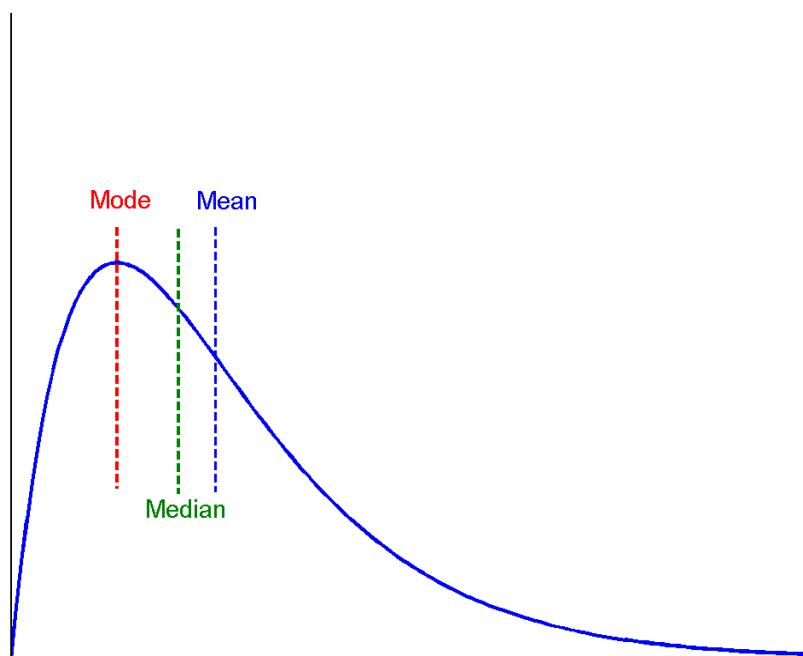
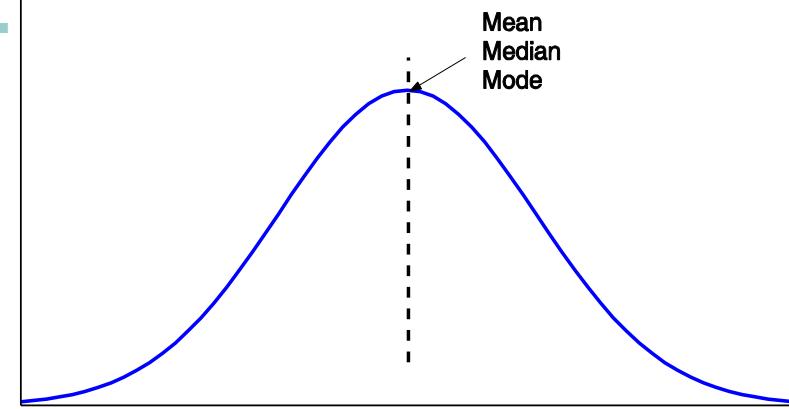
- Motivation
 - To better understand the data: central tendency, variation and spread
- Data dispersion characteristics
 - median, max, min, quantiles, outliers, variance, etc.
- Numerical dimensions correspond to sorted intervals
 - Data dispersion: analyzed with multiple granularities of precision
 - Boxplot or quantile analysis on sorted intervals
- Dispersion analysis on computed measures
 - Folding measures into numerical dimensions
 - Boxplot or quantile analysis on the transformed cube

Measuring the Central Tendency

- Mean (algebraic measure) (sample vs. population): $\bar{x} = \frac{1}{n} \sum_{i=1}^n x_i$ $\mu = \frac{\sum x}{N}$
 - Weighted arithmetic mean:
 - Trimmed mean: chopping extreme values
$$\bar{x} = \frac{\sum_{i=1}^n w_i x_i}{\sum_{i=1}^n w_i}$$
- Median: A holistic measure
 - Middle value if odd number of values, or average of the middle two values otherwise
 - Estimated by interpolation (for *grouped data*):
$$median = L_1 + \left(\frac{n/2 - (\sum f)l}{f_{median}} \right) c$$
- Mode
 - Value that occurs most frequently in the data
 - Unimodal, bimodal, trimodal
 - Empirical formula: $mean - mode = 3 \times (mean - median)$

Symmetric vs. Skewed Data

- Median, mean and mode of symmetric, positively and negatively skewed data



Measuring the Dispersion of Data

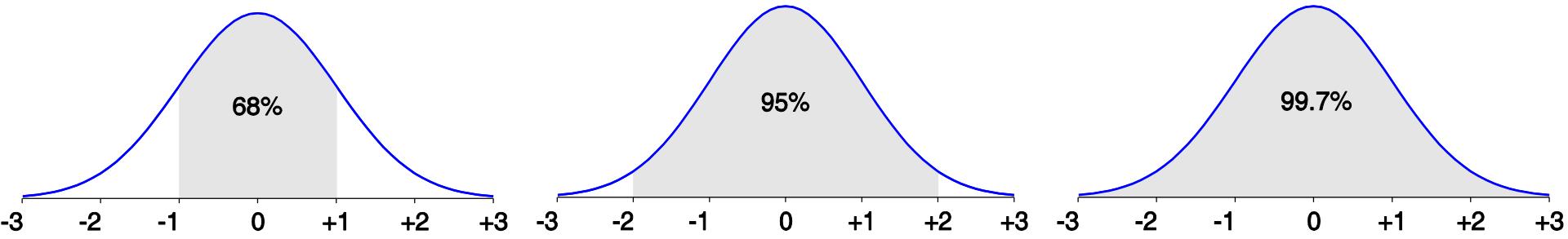
- Quartiles, outliers and boxplots
 - **Quartiles**: Q_1 (25^{th} percentile), Q_3 (75^{th} percentile)
 - **Inter-quartile range**: $\text{IQR} = Q_3 - Q_1$
 - **Five number summary**: min, Q_1 , M, Q_3 , max
 - **Boxplot**: ends of the box are the quartiles, median is marked, whiskers, and plot outlier individually
 - **Outlier**: usually, a value higher/lower than $1.5 \times \text{IQR}$
- Variance and standard deviation (*sample: s, population: σ*)
 - **Variance**: (algebraic, scalable computation)

$$s^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2 = \frac{1}{n-1} \left[\sum_{i=1}^n x_i^2 - \frac{1}{n} \left(\sum_{i=1}^n x_i \right)^2 \right] \quad \sigma^2 = \frac{1}{N} \sum_{i=1}^n (x_i - \mu)^2 = \frac{1}{N} \sum_{i=1}^n x_i^2 - \mu^2$$

- **Standard deviation** s (or σ) is the square root of variance s^2 (or σ^2)

Properties of Normal Distribution Curve

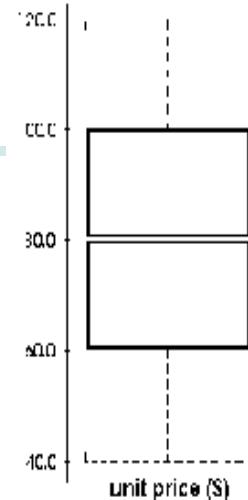
- The normal (distribution) curve
 - From $\mu-\sigma$ to $\mu+\sigma$: contains about 68% of the measurements (μ : mean, σ : standard deviation)
 - From $\mu-2\sigma$ to $\mu+2\sigma$: contains about 95% of it
 - From $\mu-3\sigma$ to $\mu+3\sigma$: contains about 99.7% of it



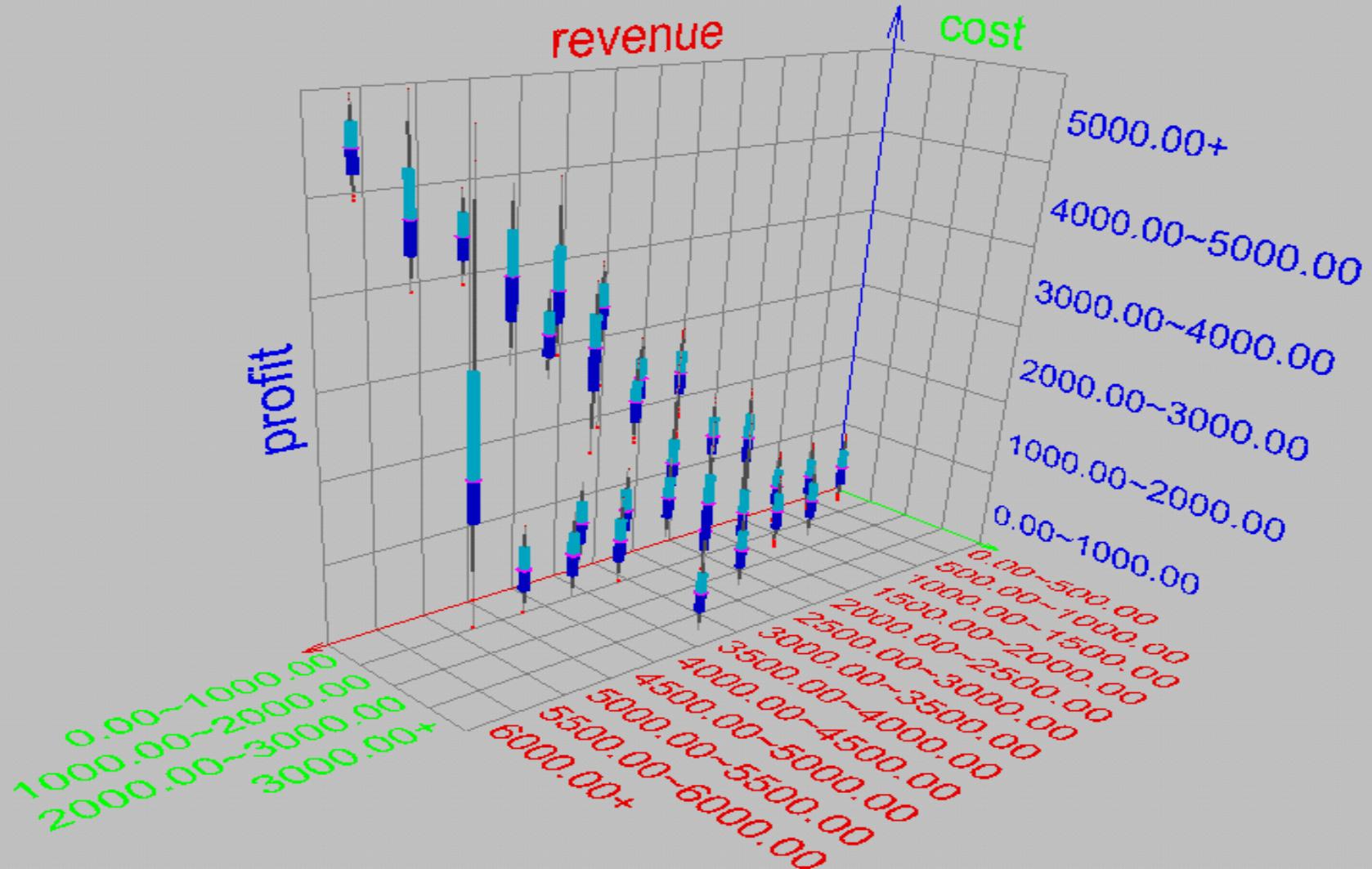
Boxplot Analysis

- Five-number summary of a distribution:
Minimum, Q1, M, Q3, Maximum

- Boxplot
 - Data is represented with a box
 - The ends of the box are at the first and third quartiles, i.e., the height of the box is IRQ
 - The median is marked by a line within the box
 - Whiskers: two lines outside the box extend to Minimum and Maximum

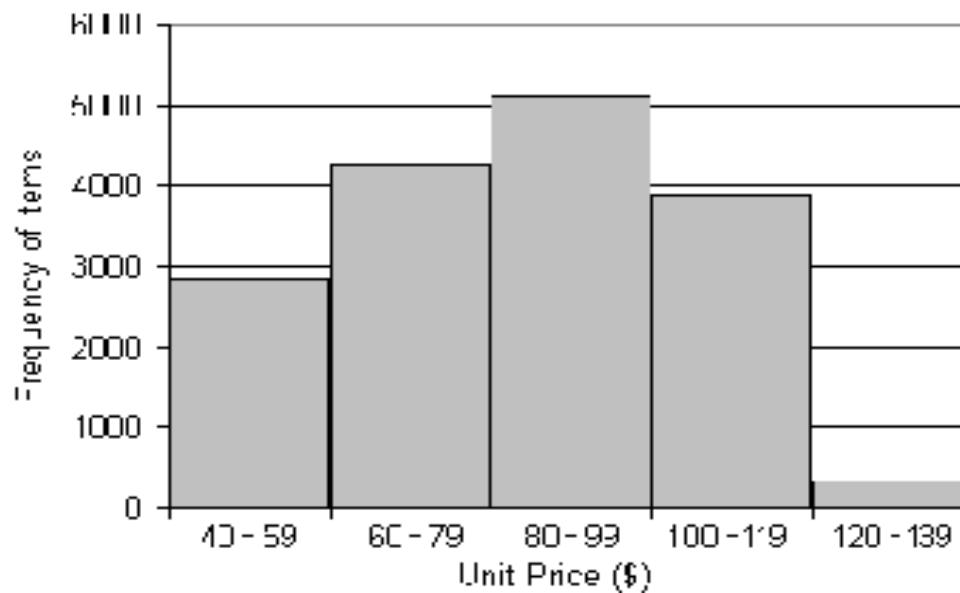


Visualization of Data Dispersion: Boxplot Analysis



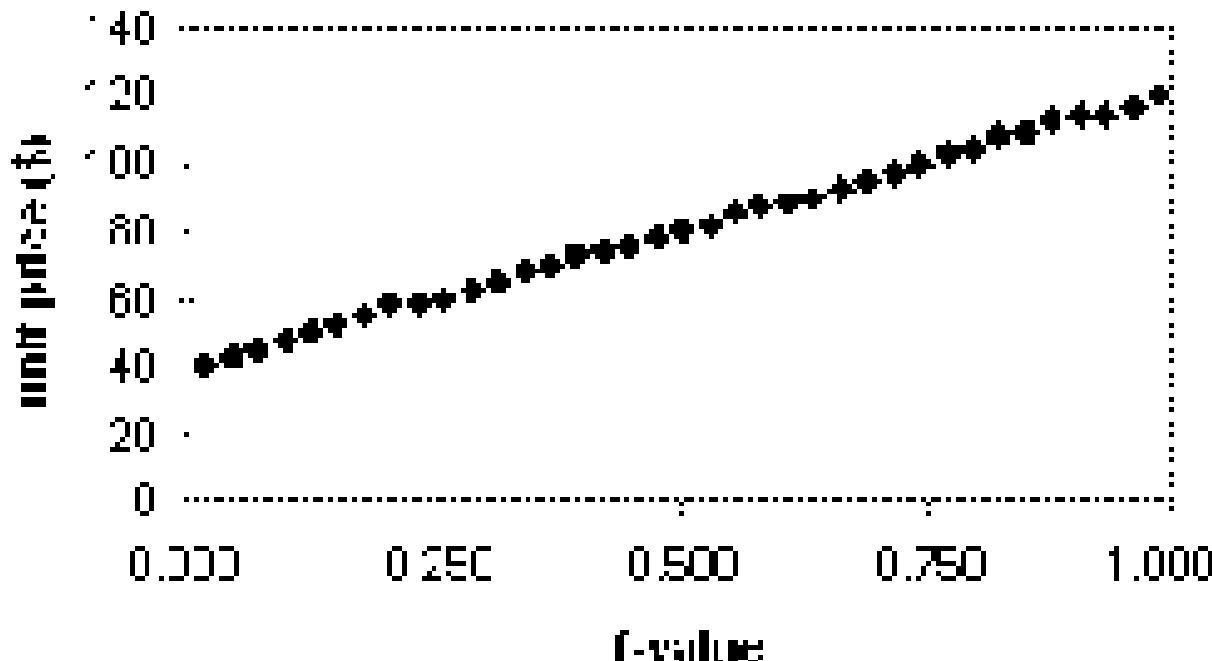
Histogram Analysis

- Graph displays of basic statistical class descriptions
 - Frequency histograms
 - A univariate graphical method
 - Consists of a set of rectangles that reflect the counts or frequencies of the classes present in the given data



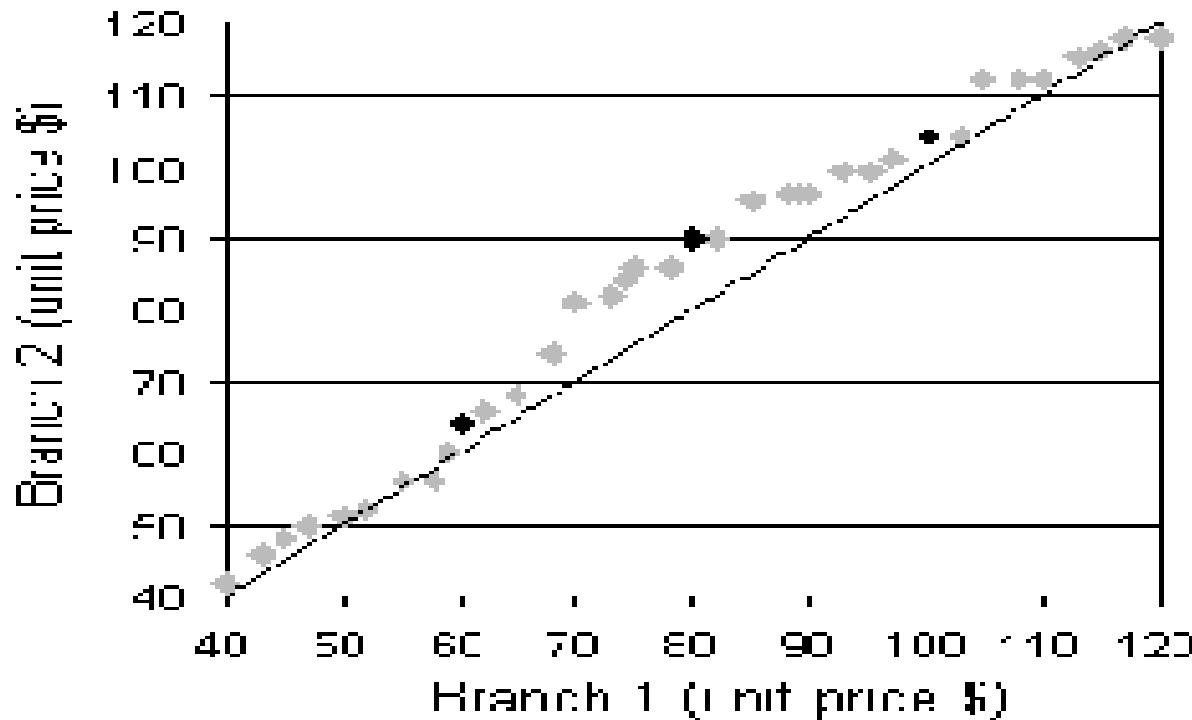
Quantile Plot

- Displays all of the data (allowing the user to assess both the overall behavior and unusual occurrences)
- Plots **quantile** information
 - For a data x_i , data sorted in increasing order, f_i indicates that approximately $100 f_i\%$ of the data are below or equal to the value x_i



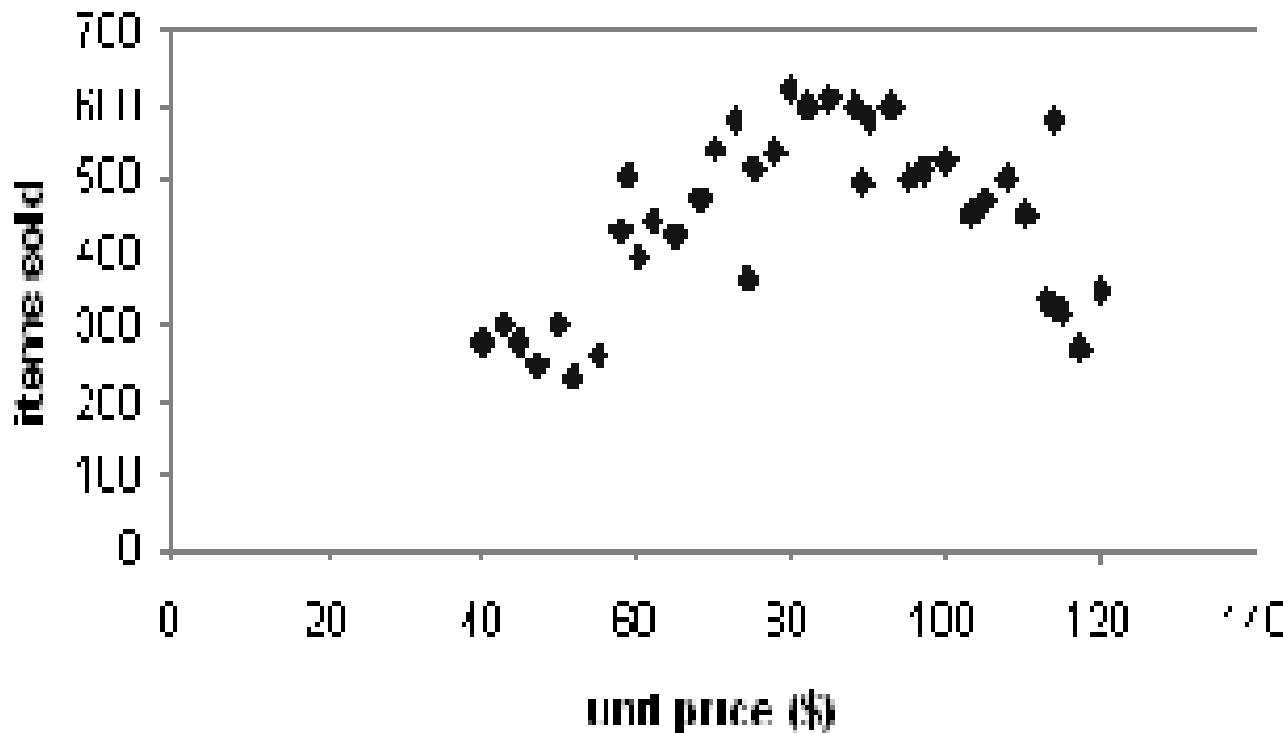
Quantile-Quantile (Q-Q) Plot

- Graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- Allows the user to view whether there is a shift in going from one distribution to another



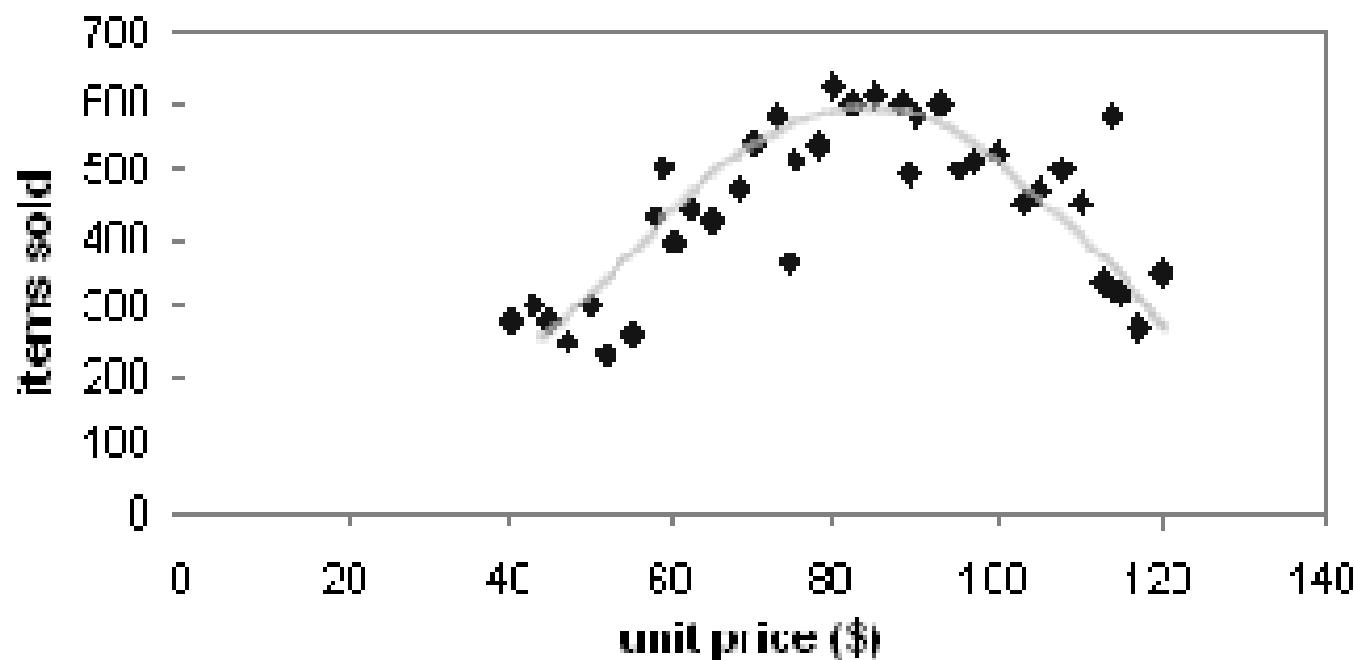
Scatter plot

- Provides a first look at bivariate data to see clusters of points, outliers, etc
- Each pair of values is treated as a pair of coordinates and plotted as points in the plane

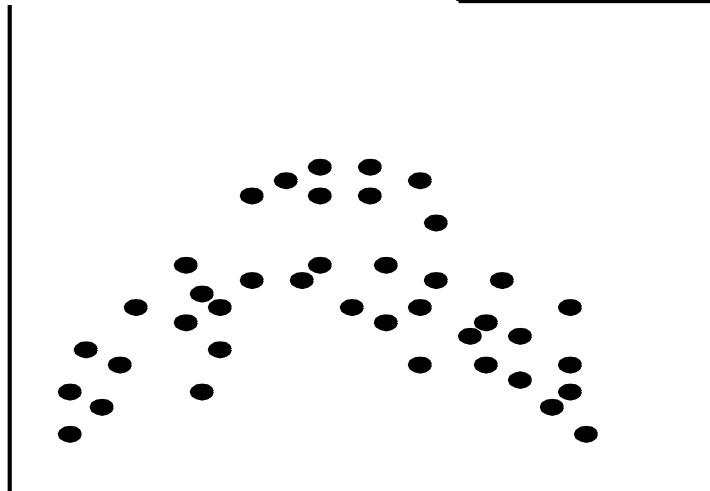
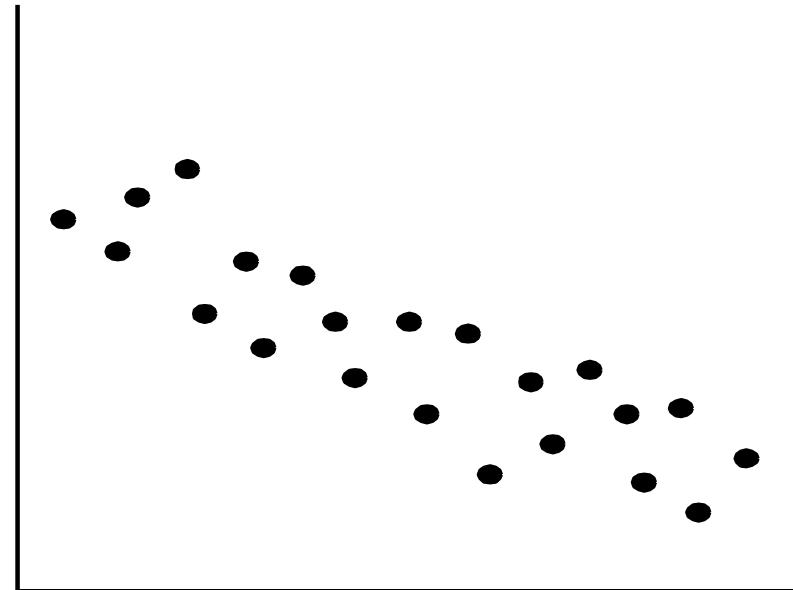
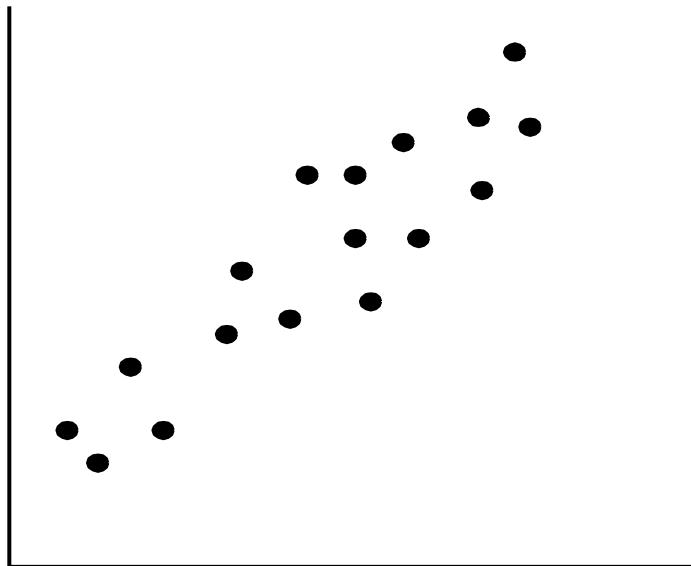


Loess Curve

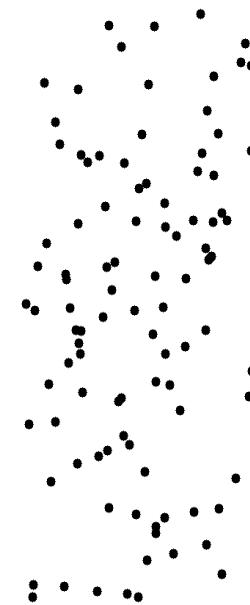
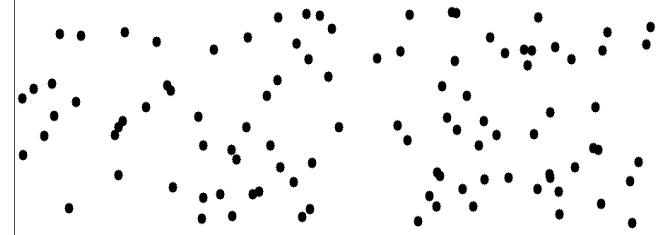
- Adds a smooth curve to a scatter plot in order to provide better perception of the pattern of dependence
- Loess curve is fitted by setting two parameters: a smoothing parameter, and the degree of the polynomials that are fitted by the regression



Positively and Negatively Correlated Data



Not Correlated Data



Graphic Displays of Basic Statistical Descriptions

- Histogram: (shown before)
- Boxplot: (covered before)
- Quantile plot: each value x_i is paired with f_i indicating that approximately $100 f_i\%$ of data are $\leq x_i$
- Quantile-quantile (q-q) plot: graphs the quantiles of one univariate distribution against the corresponding quantiles of another
- Scatter plot: each pair of values is a pair of coordinates and plotted as points in the plane
- Loess (local regression) curve: add a smooth curve to a scatter plot to provide better perception of the pattern of dependence

Chapter 2: Data Preprocessing

- Why preprocess the data?
- Descriptive data summarization
- **Data cleaning**
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Cleaning

- Importance
 - “Data cleaning is one of the three biggest problems in data warehousing”
 - “Data cleaning is the number one problem in data warehousing”
- Data cleaning tasks
 - Fill in missing values
 - Identify outliers and smooth out noisy data
 - Correct inconsistent data
 - Resolve redundancy caused by data integration

Missing Data

- Data is not always available
 - E.g., many tuples have no recorded value for several attributes, such as customer income in sales data
- Missing data may be due to
 - equipment malfunction
 - inconsistent with other recorded data and thus deleted
 - data not entered due to misunderstanding
 - certain data may not be considered important at the time of entry
 - not register history or changes of the data
- Missing data may need to be inferred.

How to Handle Missing Data?

- Ignore the tuple: usually done when class label is missing (assuming the tasks in classification—not effective when the percentage of missing values per attribute varies considerably).
- Fill in the missing value manually: tedious + infeasible?
- Fill in it automatically with
 - a global constant : e.g., “unknown”, a new class?!
 - the attribute mean
 - the attribute mean for all samples belonging to the same class: smarter
 - the most probable value: inference-based such as Bayesian formula or decision tree

Noisy Data

- Noise: random error or variance in a measured variable
- Incorrect attribute values may due to
 - faulty data collection instruments
 - data entry problems
 - data transmission problems
 - technology limitation
 - inconsistency in naming convention
- Other data problems which requires data cleaning
 - duplicate records
 - incomplete data
 - inconsistent data

How to Handle Noisy Data?

- Binning
 - first sort data and partition into (equal-frequency) bins
 - then one can smooth by bin means, smooth by bin median, smooth by bin boundaries, etc.
- Regression
 - smooth by fitting the data into regression functions
- Clustering
 - detect and remove outliers
- Combined computer and human inspection
 - detect suspicious values and check by human (e.g., deal with possible outliers)

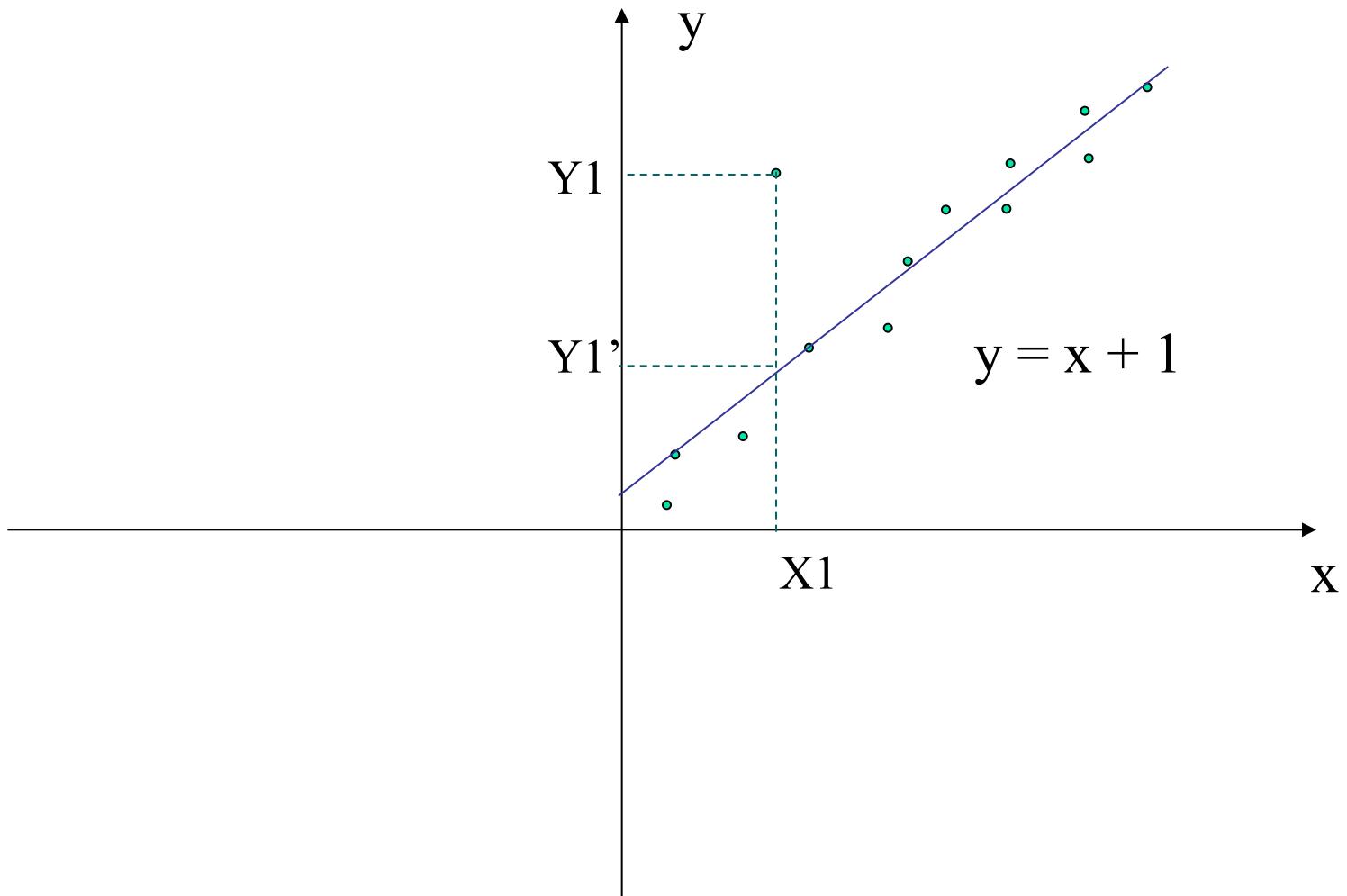
Simple Discretization Methods: Binning

- **Equal-width** (distance) partitioning
 - Divides the range into N intervals of equal size: uniform grid
 - if A and B are the lowest and highest values of the attribute, the width of intervals will be: $W = (B - A)/N$.
 - The most straightforward, but outliers may dominate presentation
 - Skewed data is not handled well
- **Equal-depth** (frequency) partitioning
 - Divides the range into N intervals, each containing approximately same number of samples
 - Good data scaling
 - Managing categorical attributes can be tricky

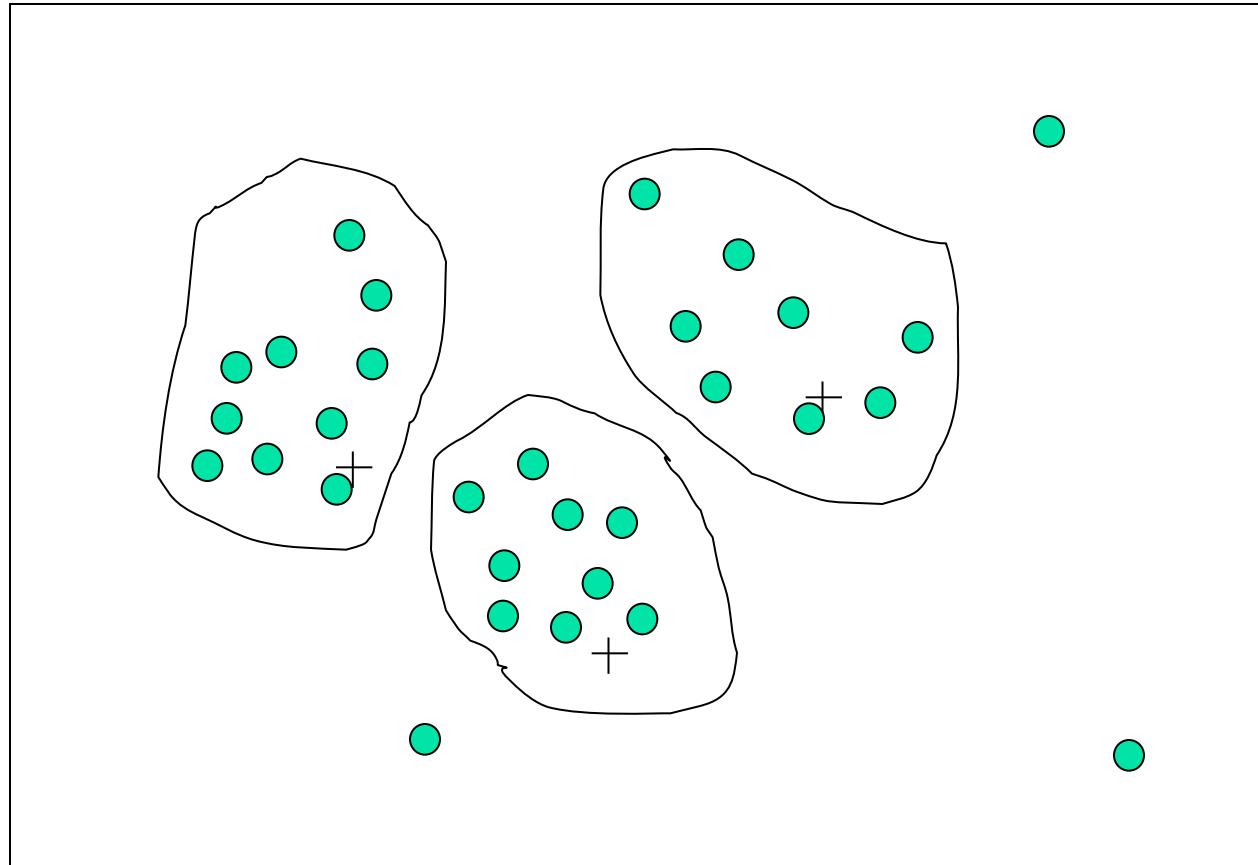
Binning Methods for Data Smoothing

- Sorted data for price (in dollars): 4, 8, 9, 15, 21, 21, 24, 25, 26, 28, 29, 34
 - * Partition into equal-frequency (equi-depth) bins:
 - Bin 1: 4, 8, 9, 15
 - Bin 2: 21, 21, 24, 25
 - Bin 3: 26, 28, 29, 34
 - * Smoothing by bin means:
 - Bin 1: 9, 9, 9, 9
 - Bin 2: 23, 23, 23, 23
 - Bin 3: 29, 29, 29, 29
 - * Smoothing by bin boundaries:
 - Bin 1: 4, 4, 4, 15
 - Bin 2: 21, 21, 25, 25
 - Bin 3: 26, 26, 26, 34

Regression



Cluster Analysis



Data Cleaning as a Process

- Data discrepancy detection
 - Use metadata (e.g., domain, range, dependency, distribution)
 - Check field overloading
 - Check uniqueness rule, consecutive rule and null rule
 - Use commercial tools
 - Data scrubbing: use simple domain knowledge (e.g., postal code, spell-check) to detect errors and make corrections
 - Data auditing: by analyzing data to discover rules and relationship to detect violators (e.g., correlation and clustering to find outliers)
- Data migration and integration
 - Data migration tools: allow transformations to be specified
 - ETL (Extraction/Transformation>Loading) tools: allow users to specify transformations through a graphical user interface
- Integration of the two processes
 - Iterative and interactive (e.g., Potter's Wheels)

Chapter 2: Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Integration

- Data integration:
 - Combines data from multiple sources into a coherent store
- Schema integration: e.g., A.cust-id \equiv B.cust-#
 - Integrate metadata from different sources
- Entity identification problem:
 - Identify real world entities from multiple data sources, e.g., Bill Clinton = William Clinton
- Detecting and resolving data value conflicts
 - For the same real world entity, attribute values from different sources are different
 - Possible reasons: different representations, different scales, e.g., metric vs. British units

Handling Redundancy in Data Integration

- Redundant data occur often when integration of multiple databases
 - *Object identification:* The same attribute or object may have different names in different databases
 - *Derivable data:* One attribute may be a “derived” attribute in another table, e.g., annual revenue
- Redundant attributes may be able to be detected by *correlation analysis*
- Careful integration of the data from multiple sources may help reduce/avoid redundancies and inconsistencies and improve mining speed and quality

Correlation Analysis (Numerical Data)

- Correlation coefficient (also called Pearson's product moment coefficient)

$$r_{A,B} = \frac{\sum (A - \bar{A})(B - \bar{B})}{(n-1)\sigma_A\sigma_B} = \frac{\sum (AB) - n\bar{A}\bar{B}}{(n-1)\sigma_A\sigma_B}$$

where n is the number of tuples, \bar{A} and \bar{B} are the respective means of A and B, σ_A and σ_B are the respective standard deviation of A and B, and $\Sigma(AB)$ is the sum of the AB cross-product.

- If $r_{A,B} > 0$, A and B are positively correlated (A's values increase as B's). The higher, the stronger correlation.
- $r_{A,B} = 0$: independent; $r_{A,B} < 0$: negatively correlated

Correlation Analysis (Categorical Data)

- χ^2 (chi-square) test

$$\chi^2 = \sum \frac{(Observed - Expected)^2}{Expected}$$

- The larger the χ^2 value, the more likely the variables are related
- The cells that contribute the most to the χ^2 value are those whose actual count is very different from the expected count
- Correlation does not imply causality
 - # of hospitals and # of car-theft in a city are correlated
 - Both are causally linked to the third variable: population

Chi-Square Calculation: An Example

	Play chess	Not play chess	Sum (row)
Like science fiction	250(90)	200(360)	450
Not like science fiction	50(210)	1000(840)	1050
Sum(col.)	300	1200	1500

- χ^2 (chi-square) calculation (numbers in parenthesis are expected counts calculated based on the data distribution in the two categories)

$$\chi^2 = \frac{(250-90)^2}{90} + \frac{(50-210)^2}{210} + \frac{(200-360)^2}{360} + \frac{(1000-840)^2}{840} = 507.93$$

- It shows that `like_science_fiction` and `play_chess` are correlated in the group

Data Transformation

- Smoothing: remove noise from data
- Aggregation: summarization, data cube construction
- Generalization: concept hierarchy climbing
- Normalization: scaled to fall within a small, specified range
 - min-max normalization
 - z-score normalization
 - normalization by decimal scaling
- Attribute/feature construction
 - New attributes constructed from the given ones

Data Transformation: Normalization

- Min-max normalization: to $[new_min_A, new_max_A]$

$$v' = \frac{v - min_A}{max_A - min_A} (new_max_A - new_min_A) + new_min_A$$

- Ex. Let income range \$12,000 to \$98,000 normalized to [0.0, 1.0]. Then \$73,000 is mapped to $\frac{73,600 - 12,000}{98,000 - 12,000}(1.0 - 0) + 0 = 0.716$
- Z-score normalization (μ : mean, σ : standard deviation):

$$v' = \frac{v - \mu_A}{\sigma_A}$$

- Ex. Let $\mu = 54,000$, $\sigma = 16,000$. Then $\frac{73,600 - 54,000}{16,000} = 1.225$
- Normalization by decimal scaling

$$v' = \frac{v}{10^j} \quad \text{Where } j \text{ is the smallest integer such that } \text{Max}(|v'|) < 1$$

Chapter 2: Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Data Reduction Strategies

- Why data reduction?
 - A database/data warehouse may store terabytes of data
 - Complex data analysis/mining may take a very long time to run on the complete data set
- Data reduction
 - Obtain a reduced representation of the data set that is much smaller in volume but yet produce the same (or almost the same) analytical results
- Data reduction strategies
 - Data cube aggregation: aggregation operations
 - Attribute subset selection: remove unimportant attributes
 - Dimensionality reduction: encoding mechanisms used
 - Numerosity reduction : fit data into models
 - Discretization and concept hierarchy generation

Data Cube Aggregation

- The lowest level of a data cube (base cuboid)
 - The aggregated data for an **individual entity of interest**
 - E.g., a customer in a phone calling data warehouse
- Multiple levels of aggregation in data cubes
 - Further reduce the size of data to deal with
- Reference appropriate levels
 - Use the smallest representation which is enough to solve the task
- Queries regarding aggregated information should be answered using data cube, when possible

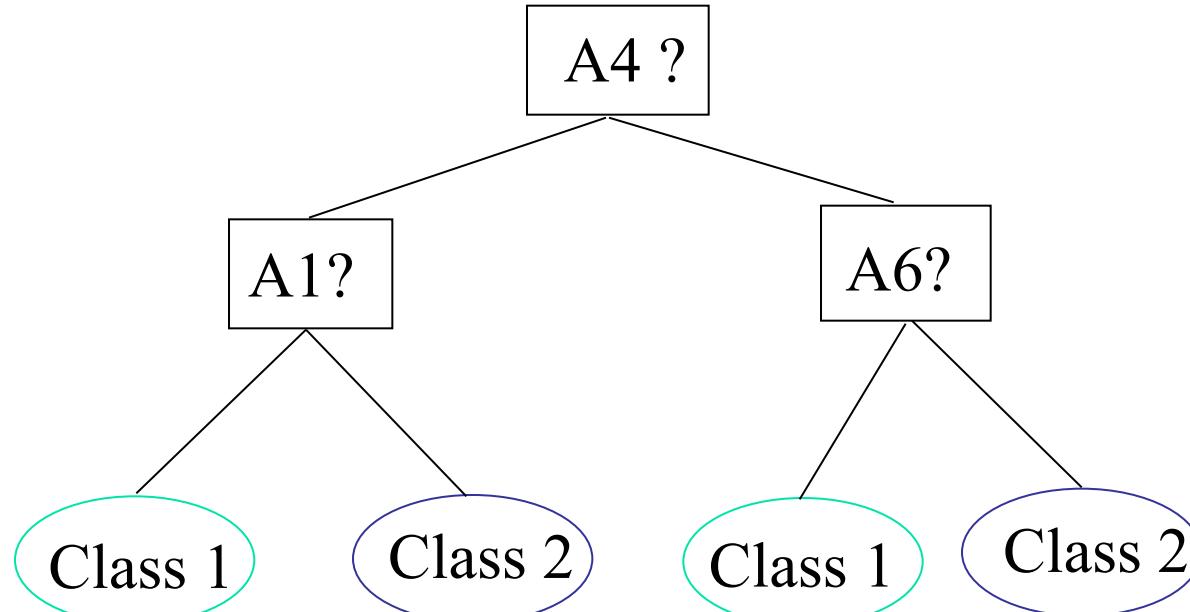
Attribute Subset Selection

- Feature selection (i.e., attribute subset selection):
 - Select a minimum set of features such that the probability distribution of different classes given the values for those features is as close as possible to the original distribution given the values of all features
 - reduce # of patterns in the patterns, easier to understand
- Heuristic methods (due to exponential # of choices):
 - Step-wise forward selection
 - Step-wise backward elimination
 - Combining forward selection and backward elimination
 - Decision-tree induction

Example of Decision Tree Induction

Initial attribute set:

$\{A_1, A_2, A_3, A_4, A_5, A_6\}$



-----> Reduced attribute set: $\{A_1, A_4, A_6\}$

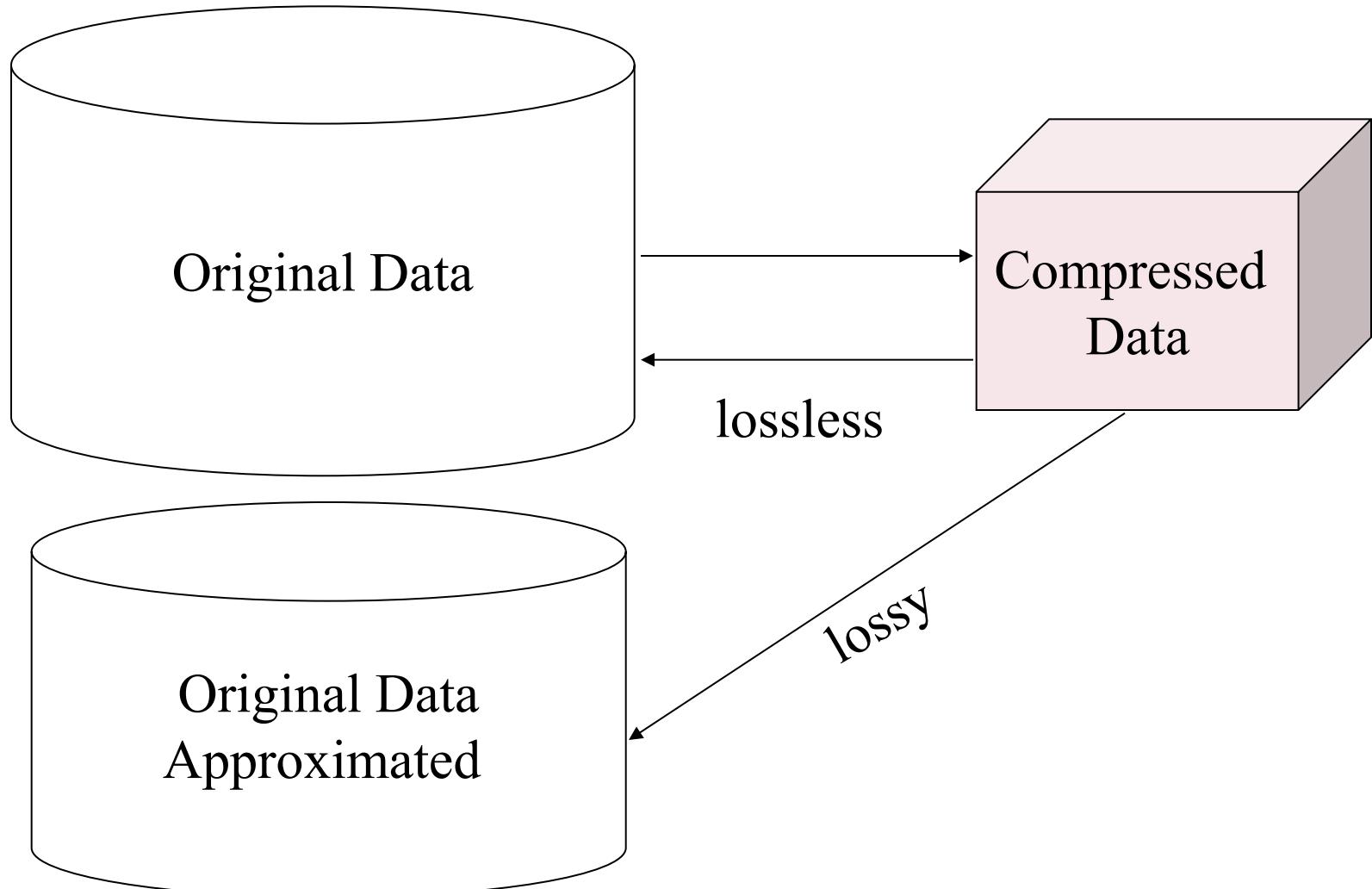
Heuristic Feature Selection Methods

- There are 2^d possible sub-features of d features
- Several heuristic feature selection methods:
 - Best single features under the feature independence assumption: choose by significance tests
 - Best step-wise feature selection:
 - The best single-feature is picked first
 - Then next best feature condition to the first, ...
 - Step-wise feature elimination:
 - Repeatedly eliminate the worst feature
 - Best combined feature selection and elimination
 - Optimal branch and bound:
 - Use feature elimination and backtracking

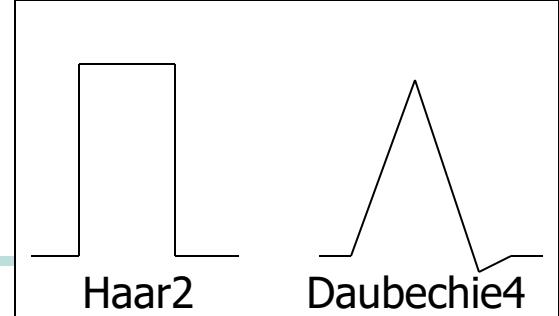
Data Compression

- String compression
 - There are extensive theories and well-tuned algorithms
 - Typically lossless
 - But only limited manipulation is possible without expansion
- Audio/video compression
 - Typically lossy compression, with progressive refinement
 - Sometimes small fragments of signal can be reconstructed without reconstructing the whole
- Time sequence is not audio
 - Typically short and vary slowly with time

Data Compression

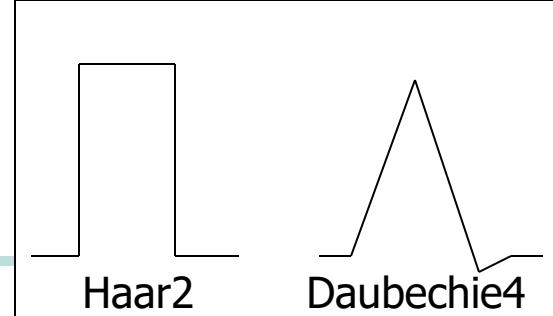


Dimensionality Reduction: Wavelet Transformation



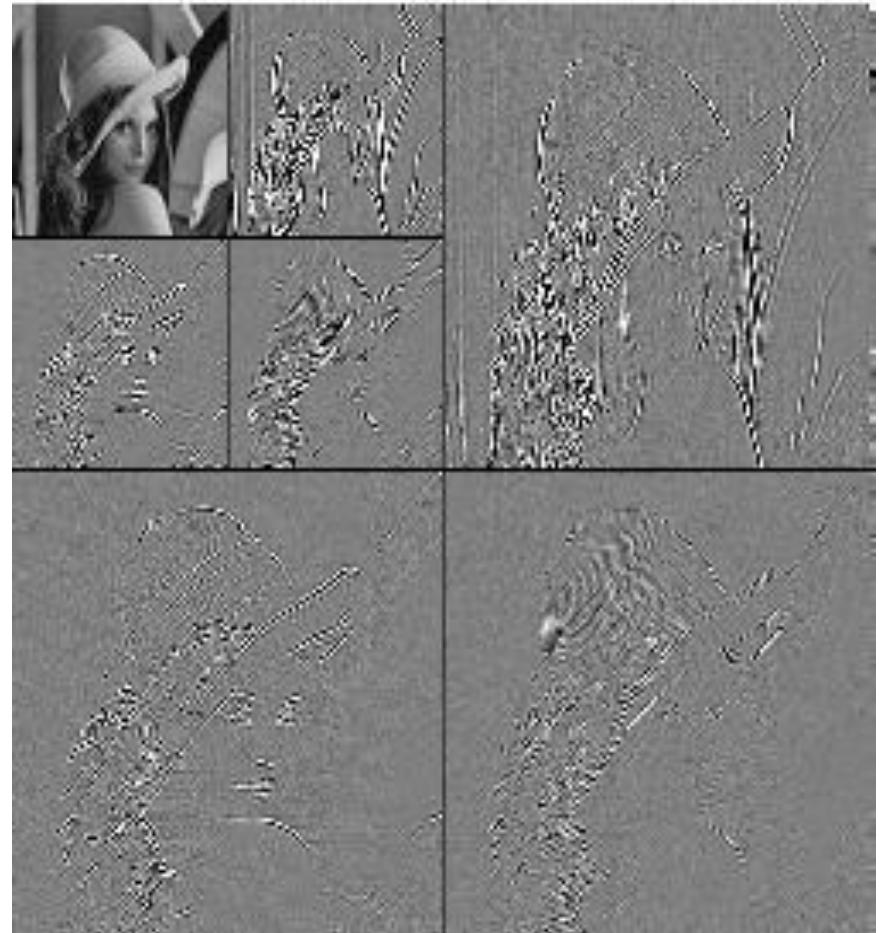
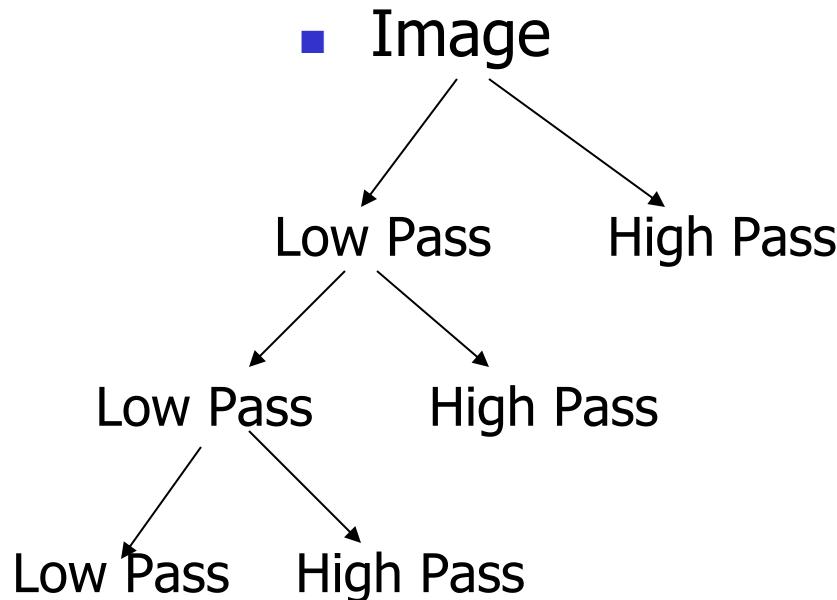
- Data encoding techniques or transformation are applied on the original data to obtain a reduced or compressed representation of the data.
- Reconstructed data can be lossy or lossless
- Lossy dimensionality reduction methods:
 - Wavelet transforms(DWT)
 - Haar transform
 - Principle component analysis(PCA)
 - K-L method

Dimensionality Reduction: Wavelet Transformation



- Discrete wavelet transform (DWT): linear signal processing, multi-resolutonal analysis
- Compressed approximation: store only a small fraction of the strongest of the wavelet coefficients
- Similar to discrete Fourier transform (DFT), but better lossy compression, localized in space
- Method:
 - Length, L , must be an integer power of 2 (padding with 0's, when necessary)
 - Each transform has 2 functions: smoothing, difference
 - Applies to pairs of data, resulting in two set of data of length $L/2$
 - Applies two functions recursively, until reaches the desired length

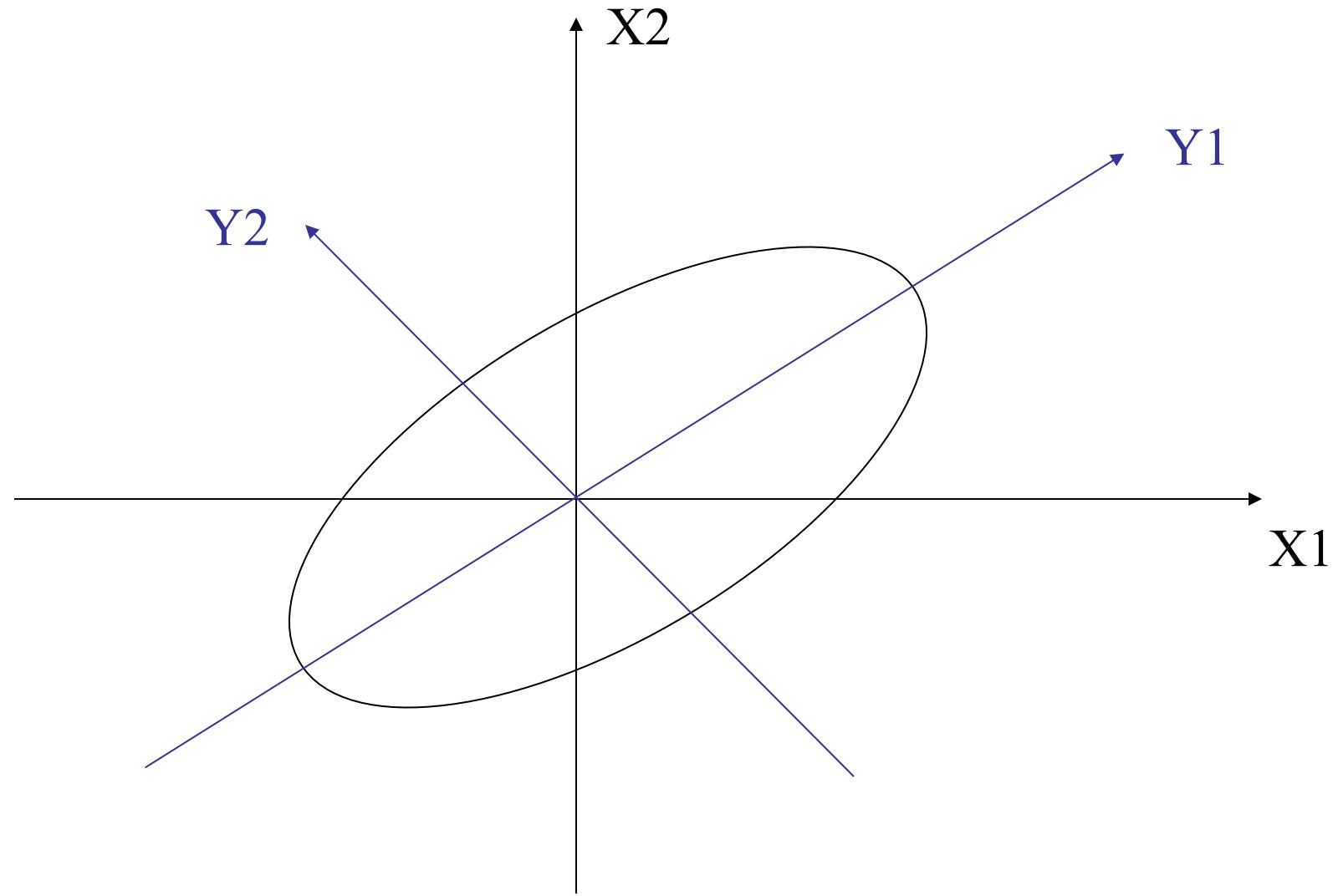
DWT for Image Compression



Dimensionality Reduction: Principal Component Analysis (PCA)

- Given N data vectors from n -dimensions, find $k \leq n$ orthogonal vectors (*principal components*) that can be best used to represent data
- Steps
 - Normalize input data: Each attribute falls within the same range
 - Compute k orthonormal (unit) vectors, i.e., *principal components*
 - Each input data (vector) is a linear combination of the k principal component vectors
 - The principal components are sorted in order of decreasing “significance” or strength
 - Since the components are sorted, the size of the data can be reduced by eliminating the weak components, i.e., those with low variance. (i.e., using the strongest principal components, it is possible to reconstruct a good approximation of the original data
- Works for numeric data only
- Used when the number of dimensions is large

Principal Component Analysis



Numerosity Reduction

- Reduce data volume by choosing alternative, smaller forms of data representation
- **Parametric methods**
 - Assume the data fits some model, estimate model parameters, store only the parameters, and discard the data (except possible outliers)
 - Example: **Regression and Log-linear models**—obtain value at a point in m-D space as the product on appropriate marginal subspaces
- **Non-parametric methods**
 - Do not assume models
 - Major families: **histograms, clustering, sampling**

Data Reduction Method (1): Regression and Log-Linear Models

- **Linear regression:** Data are modeled to fit a **straight line**

$$y = wx + b$$

Where, y and x → numerical database attributes

w and b → regression coefficient

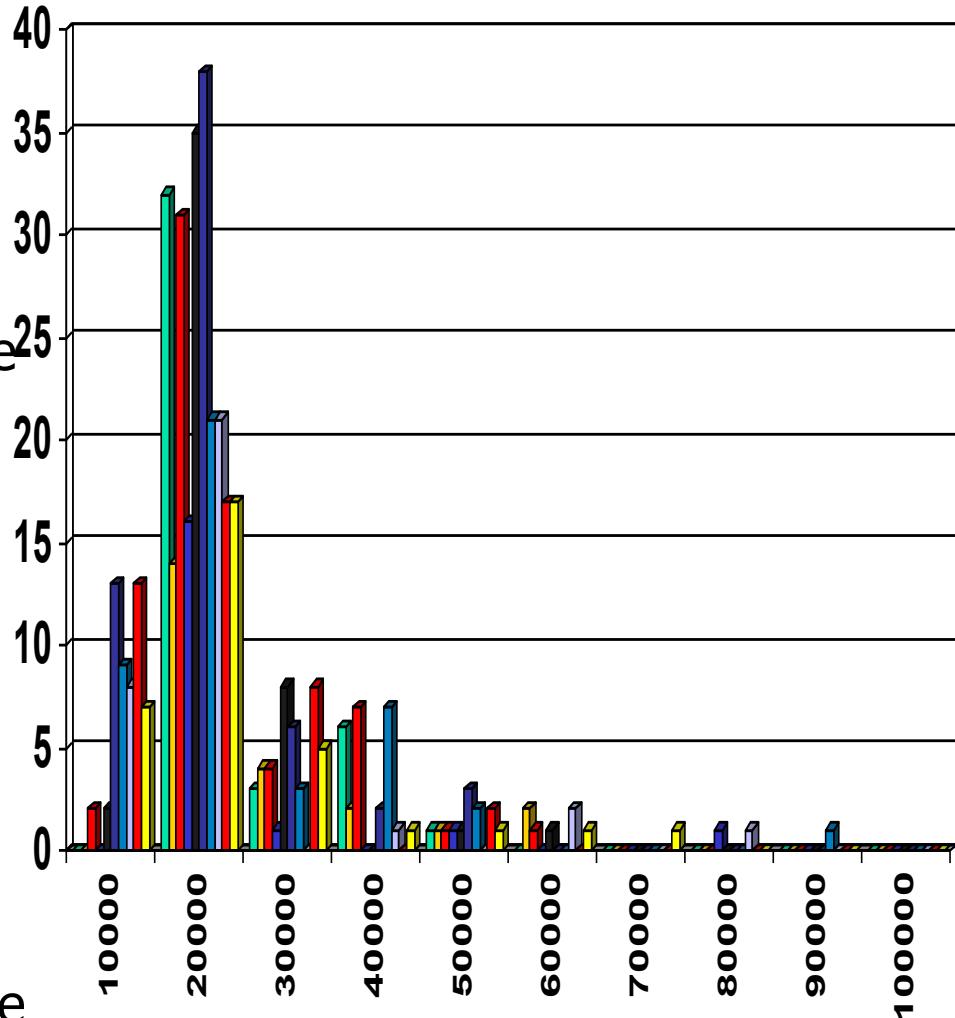
- **Multiple regression:** extension of linear regression method where y is modeled as a linear function of **two or more predictor variable**
- **Log-linear model:** estimate the **probability of each point** in a multidimensional space for a set of discretized attributes, based on the smaller subset of dimensional combinations

Regress Analysis and Log-Linear Models

- Linear regression: $Y = w X + b$
 - Two regression coefficients, w and b , specify the line and are to be estimated by using the data at hand
 - Using the least squares criterion to the known values of $Y_1, Y_2, \dots, X_1, X_2, \dots$
- Multiple regression: $Y = b_0 + b_1 X_1 + b_2 X_2$.
 - Many nonlinear functions can be transformed into the above
- Log-linear models:
 - The multi-way table of joint probabilities is approximated by a product of lower-order tables
 - Probability: $p(a, b, c, d) = \alpha_{ab} \beta_{ac} \gamma_{ad} \delta_{bcd}$

Data Reduction Method (2): Histograms

- Histogram for an attribute A → partition data distribution of A into disjoint subsets or buckets.
- Partitioning rules:
 - Equal-width: equal bucket range
 - Equal-frequency (or equal-depth)
 - V-optimal: with the least *histogram variance* (weighted sum of the original values that each bucket represents)
 - MaxDiff: set bucket boundary between each pair for pairs have the $\beta-1$ largest differences



-
- List of prices of commonly sold items at AllElectronics(rounded to nearest dollar)
 - 1,1,5,5,5,5,8,8,10,10,10,10,10,12,14,14,14,14,15,15,15,15,15

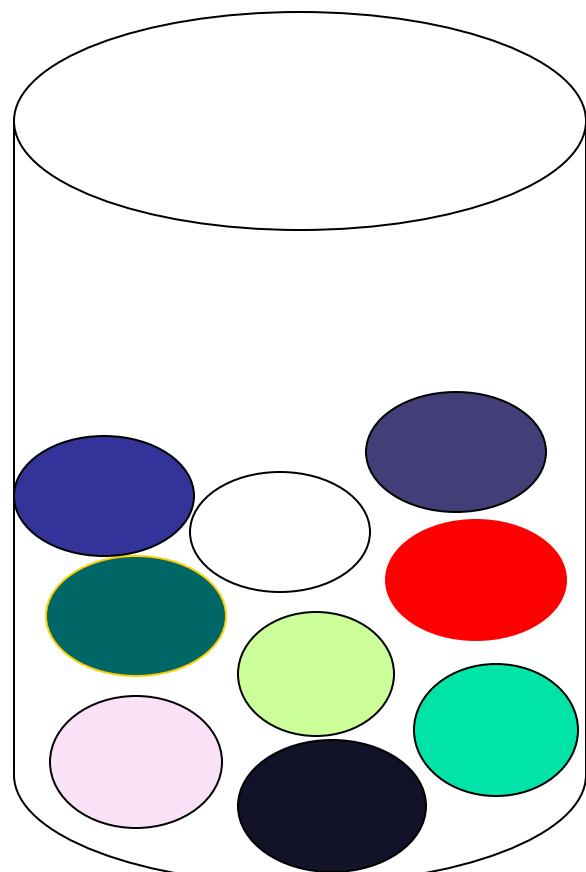
Data Reduction Method (3): Clustering

- Partition data set into clusters based on similarity, and store cluster representation (e.g., centroid and diameter) only
- Can be very effective if data is clustered but not if data is “smeared”
- Can have hierarchical clustering and be stored in multi-dimensional index tree structures
- There are many choices of clustering definitions and clustering algorithms
- Cluster analysis will be studied in depth in Chapter 7

Data Reduction Method (4): Sampling

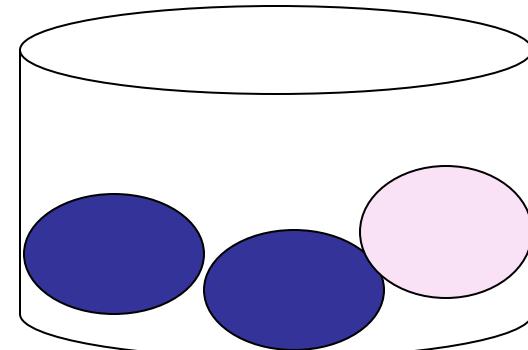
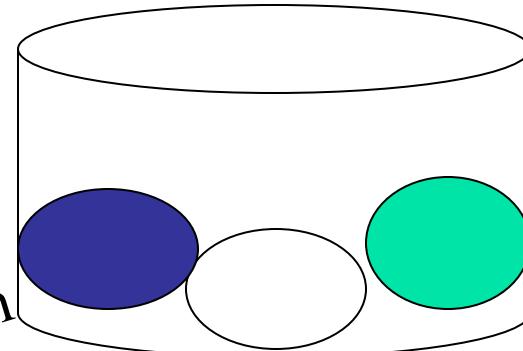
- Sampling: obtaining a small sample s to represent the whole data set N
- Allow a mining algorithm to run in complexity that is potentially sub-linear to the size of the data
- Choose a **representative** subset of the data
 - Simple random sampling may have very poor performance in the presence of skew
- Develop adaptive sampling methods
 - Stratified sampling:
 - Approximate the percentage of each class (or subpopulation of interest) in the overall database
 - Used in conjunction with skewed data
- Note: Sampling may not reduce database I/Os (page at a time)

Sampling: with or without Replacement



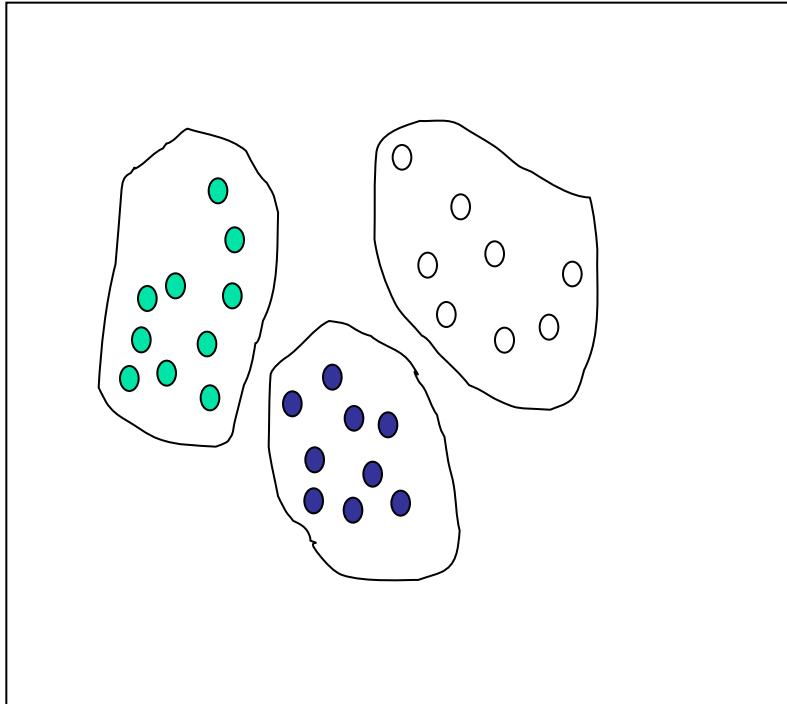
SRSWOR
(simple random
sample without
replacement)

SRSWR

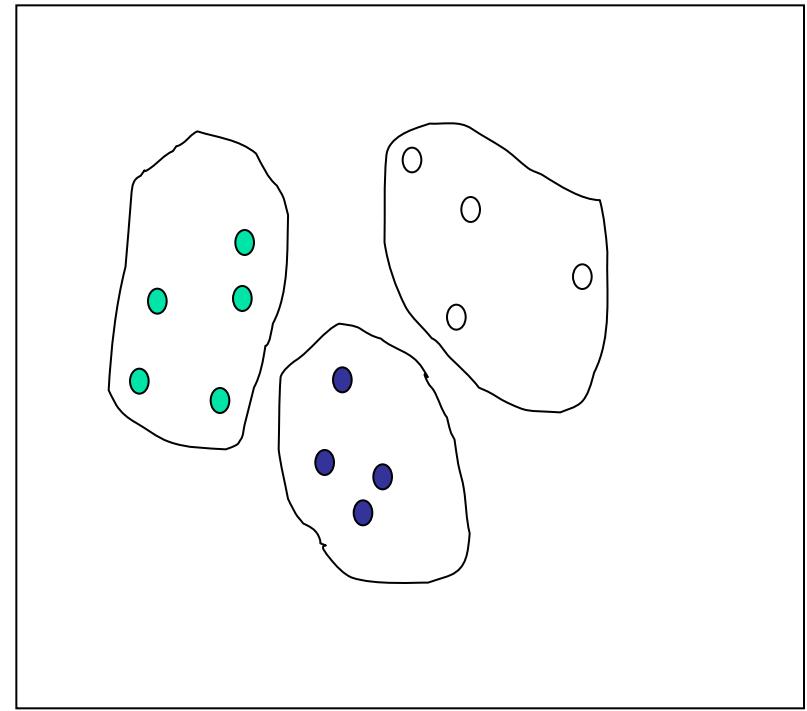


Sampling: Cluster or Stratified Sampling

Raw Data



Cluster/Stratified Sample



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- Summary

Discretization

- Three types of attributes:
 - Nominal — values from an unordered set, e.g., color, profession
 - Ordinal — values from an ordered set, e.g., military or academic rank
 - Continuous — real numbers, e.g., integer or real numbers
- Discretization:
 - Divide the range of a continuous attribute into intervals
 - Some classification algorithms only accept categorical attributes.
 - Reduce data size by discretization
 - Prepare for further analysis

Discretization and Concept Hierarchy

- Discretization
 - Reduce the number of values for a given continuous attribute by dividing the range of the attribute into intervals
 - Interval labels can then be used to replace actual data values
 - Supervised vs. unsupervised
 - Split (top-down) vs. merge (bottom-up)
 - Discretization can be performed recursively on an attribute
- Concept hierarchy formation
 - Recursively reduce the data by collecting and replacing low level concepts (such as numeric values for age) by higher level concepts (such as young, middle-aged, or senior)

Discretization and Concept Hierarchy Generation for Numeric Data

- Typical methods: All the methods can be applied recursively
 - Binning (covered above)
 - Top-down split, unsupervised,
 - Histogram analysis (covered above)
 - Top-down split, unsupervised
 - Clustering analysis (covered above)
 - Either top-down split or bottom-up merge, unsupervised
 - Entropy-based discretization: supervised, top-down split
 - Interval merging by χ^2 Analysis: unsupervised, bottom-up merge
 - Segmentation by natural partitioning: top-down split, unsupervised

Entropy-Based Discretization

- Given a set of samples S , if S is partitioned into two intervals S_1 and S_2 using boundary T , the information gain after partitioning is

$$I(S, T) = \frac{|S_1|}{|S|} Entropy(S_1) + \frac{|S_2|}{|S|} Entropy(S_2)$$

- Entropy is calculated based on class distribution of the samples in the set. Given m classes, the entropy of S_1 is

$$Entropy(S_1) = -\sum_{i=1}^m p_i \log_2(p_i)$$

where p_i is the probability of class i in S_1

- The boundary that minimizes the entropy function over all possible boundaries is selected as a binary discretization
- The process is recursively applied to partitions obtained until some stopping criterion is met
- Such a boundary may reduce data size and improve classification accuracy

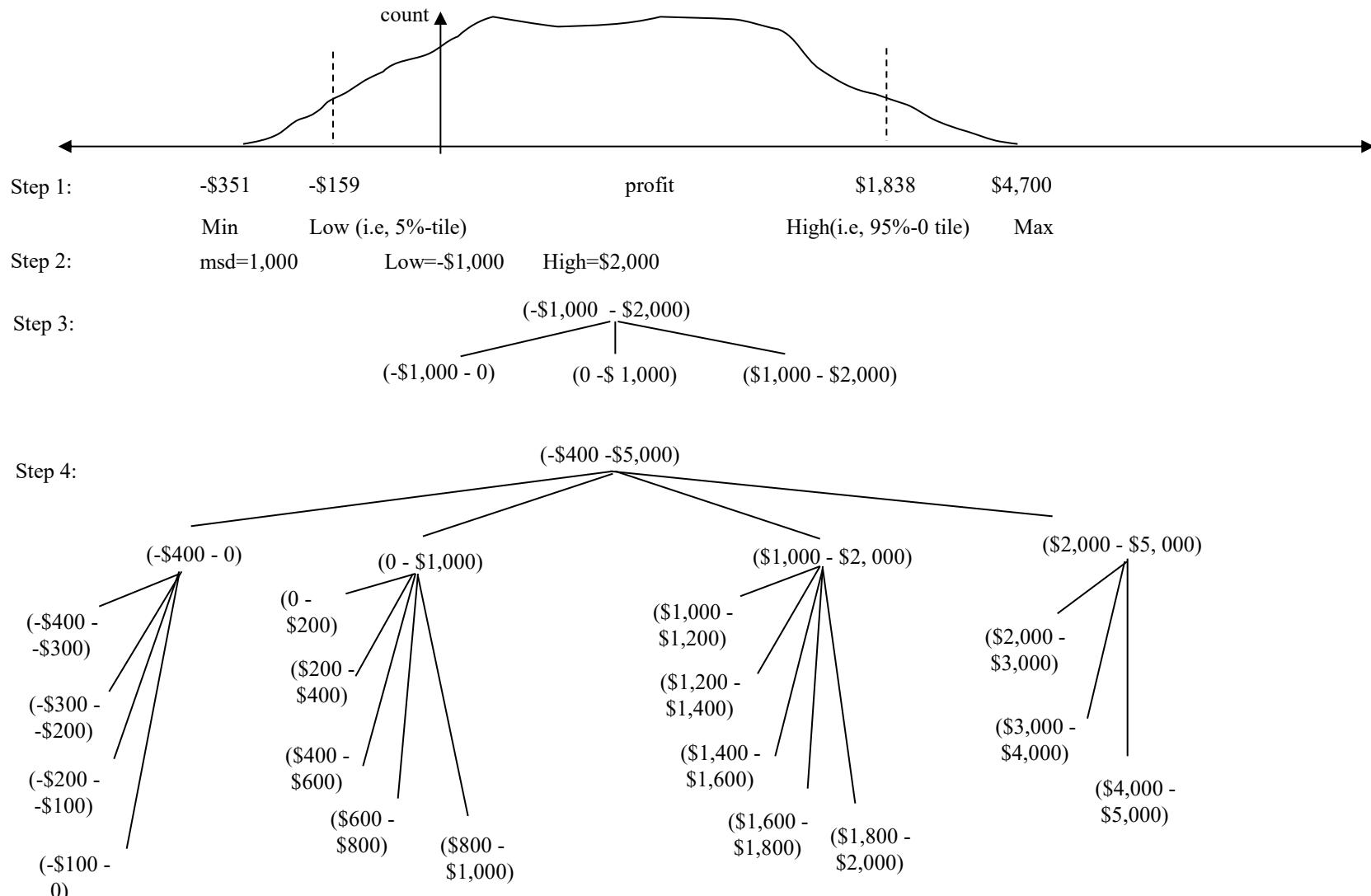
Interval Merge by χ^2 Analysis

- Merging-based (bottom-up) vs. splitting-based methods
- Merge: Find the best neighboring intervals and merge them to form larger intervals recursively
- ChiMerge [Kerber AAAI 1992, See also Liu et al. DMKD 2002]
 - Initially, each distinct value of a numerical attr. A is considered to be one interval
 - χ^2 tests are performed for every pair of adjacent intervals
 - Adjacent intervals with the least χ^2 values are merged together, since low χ^2 values for a pair indicate similar class distributions
 - This merge process proceeds recursively until a predefined stopping criterion is met (such as significance level, max-interval, max inconsistency, etc.)

Segmentation by Natural Partitioning

- A simply 3-4-5 rule can be used to segment numeric data into relatively uniform, “natural” intervals.
 - If an interval covers 3, 6, 7 or 9 distinct values at the most significant digit, partition the range into 3 equi-width intervals
 - If it covers 2, 4, or 8 distinct values at the most significant digit, partition the range into 4 intervals
 - If it covers 1, 5, or 10 distinct values at the most significant digit, partition the range into 5 intervals

Example of 3-4-5 Rule



Concept Hierarchy Generation for Categorical Data

- Specification of a partial/total ordering of attributes explicitly at the schema level by users or experts
 - street < city < state < country
- Specification of a hierarchy for a set of values by explicit data grouping
 - {Urbana, Champaign, Chicago} < Illinois
- Specification of only a partial set of attributes
 - E.g., only street < city, not others
- Automatic generation of hierarchies (or attribute levels) by the analysis of the number of distinct values
 - E.g., for a set of attributes: {street, city, state, country}

Automatic Concept Hierarchy Generation

- Some hierarchies can be automatically generated based on the analysis of the number of distinct values per attribute in the data set
 - The attribute with the most distinct values is placed at the lowest level of the hierarchy
 - Exceptions, e.g., weekday, month, quarter, year



Chapter 2: Data Preprocessing

- Why preprocess the data?
- Data cleaning
- Data integration and transformation
- Data reduction
- Discretization and concept hierarchy generation
- **Summary**

Summary

- Data preparation or preprocessing is a big issue for both data warehousing and data mining
- Descriptive data summarization is need for quality data preprocessing
- Data preparation includes
 - Data cleaning and data integration
 - Data reduction and feature selection
 - Discretization
- A lot a methods have been developed but data preprocessing still an active area of research

References

- D. P. Ballou and G. K. Tayi. Enhancing data quality in data warehouse environments. Communications of ACM, 42:73-78, 1999
- T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley & Sons, 2003
- T. Dasu, T. Johnson, S. Muthukrishnan, V. Shkapenyuk. Mining Database Structure; Or, How to Build a Data Quality Browser. SIGMOD'02.
- H.V. Jagadish et al., Special Issue on Data Reduction Techniques. Bulletin of the Technical Committee on Data Engineering, 20(4), December 1997
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999
- E. Rahm and H. H. Do. Data Cleaning: Problems and Current Approaches. *IEEE Bulletin of the Technical Committee on Data Engineering*. Vol.23, No.4
- V. Raman and J. Hellerstein. Potters Wheel: An Interactive Framework for Data Cleaning and Transformation, VLDB'2001
- T. Redman. Data Quality: Management and Technology. Bantam Books, 1992
- Y. Wand and R. Wang. Anchoring data quality dimensions ontological foundations. Communications of ACM, 39:86-95, 1996
- R. Wang, V. Storey, and C. Firth. A framework for analysis of data quality research. IEEE Trans. Knowledge and Data Engineering, 7:623-640, 1995



Data Mining

Prof. Dr. Nizamettin AYDIN

naydin@yildiz.edu.tr

<http://www3.yildiz.edu.tr/~naydin>

Data Mining

Similarity and Dissimilarity Measures

- Outline
 - Similarity and Dissimilarity between Simple Attributes
 - Dissimilarities between Data Objects
 - Similarities between Data Objects
 - Examples of Proximity
 - Mutual Information
 - Issues in Proximity
 - Selecting the Right Proximity Measure

Similarity and Dissimilarity Measures

- Similarity and dissimilarity are important because they are used by a number of data mining techniques, such as clustering, nearest neighbor classification, and anomaly detection.
- In many cases, the initial data set is not needed once these similarities or dissimilarities have been computed.
- Such approaches can be viewed as transforming the data to a similarity (dissimilarity) space and then performing the analysis.

Similarity and Dissimilarity Measures

- Similarity measure
 - Numerical measure of how alike two data objects are.
 - Is higher when objects are more alike.
 - Often falls in the range [0,1]
- Dissimilarity measure
 - Numerical measure of how different two data objects are
 - Lower when objects are more alike
 - Minimum dissimilarity is often 0, upper limit varies
 - The term **distance** is used as a synonym for dissimilarity
- Proximity refers to a similarity or dissimilarity

Transformations

- often applied to convert a similarity to a dissimilarity, or vice versa, or to transform a proximity measure to fall within a particular range, such as $[0,1]$.
 - For instance, we may have similarities that range from 1 to 10, but the particular algorithm or software package that we want to use may be designed to work only with dissimilarities, or it may work only with similarities in the interval $[0,1]$
- Frequently, proximity measures, especially similarities, are defined or transformed to have values in the interval $[0,1]$.

Transformations

- often applied to convert a similarity to a dissimilarity, or vice versa, or to transform a proximity measure to fall within a particular range, such as $[0,1]$.
 - For instance, we may have similarities that range from 1 to 10, but the particular algorithm or software package that we want to use may be designed to work only with dissimilarities, or it may work only with similarities in the interval $[0,1]$
- Frequently, proximity measures, especially similarities, are defined or transformed to have values in the interval $[0,1]$.

Transformations

- Example:
 - If the similarities between objects range from 1 (not at all similar) to 10 (completely similar), we can make them fall within the range [0, 1] by using the transformation $s' = (s-1)/9$, where s and s' are the original and new similarity values, respectively.
- The transformation of similarities and dissimilarities to the interval [0, 1]
 - $s' = (s - s_{\min}) / (s_{\max} - s_{\min})$, where s_{\max} and s_{\min} are the maximum and minimum similarity values.
 - $d' = (d - d_{\min}) / (d_{\max} - d_{\min})$, where d_{\max} and d_{\min} are the maximum and minimum dissimilarity values.

Transformations

- However, there can be complications in mapping proximity measures to the interval $[0, 1]$ using a linear transformation.
 - If, for example, the proximity measure originally takes values in the interval $[0, \infty]$, then d_{\max} is not defined and a nonlinear transformation is needed.
 - Values will not have the same relationship to one another on the new scale.
- Consider the transformation $d = d/(1+d)$ for a dissimilarity measure that ranges from 0 to ∞ .
 - Given dissimilarities 0, 0.5, 2, 10, 100, 1000
 - Transformed dissimilarities 0, 0.33, 0.67, 0.90, 0.99, 0.999.
- Larger values on the original dissimilarity scale are compressed into the range of values near 1, but whether this is desirable depends on the application.

Similarity/Dissimilarity for Simple Attributes

- The following table shows the similarity and dissimilarity between two objects, x and y , with respect to a single, simple attribute.

Attribute Type	Dissimilarity	Similarity
Nominal	$d = \begin{cases} 0 & \text{if } x = y \\ 1 & \text{if } x \neq y \end{cases}$	$s = \begin{cases} 1 & \text{if } x = y \\ 0 & \text{if } x \neq y \end{cases}$
Ordinal	$d = x - y / (n - 1)$ (values mapped to integers 0 to $n-1$, where n is the number of values)	$s = 1 - d$
Interval or Ratio	$d = x - y $	$s = -d, s = \frac{1}{1+d}, s = e^{-d},$ $s = 1 - \frac{d - \min_d}{\max_d - \min_d}$

- Next, we consider more complicated measures of proximity between objects that involve multiple attributes:
 - dissimilarities between data objects
 - similarities between data objects.

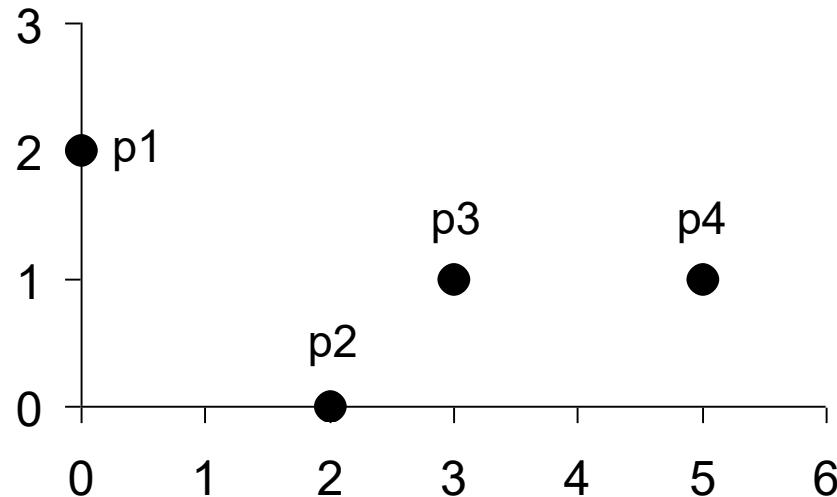
Distances - Euclidean Distance

- The Euclidean distance, d , between two points, x and y , in one-, two-, three-, or higher-dimensional space, is given by

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

- where n is the number of dimensions (attributes) and x_k and y_k are, respectively, the k^{th} attributes (components) of data objects \mathbf{x} and \mathbf{y} .
- Standardization is necessary, if scales differ.

Distances - Euclidean Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

$$d(\mathbf{x}, \mathbf{y}) = \sqrt{\sum_{k=1}^n (x_k - y_k)^2}$$

	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

Distances - Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance, and is given by

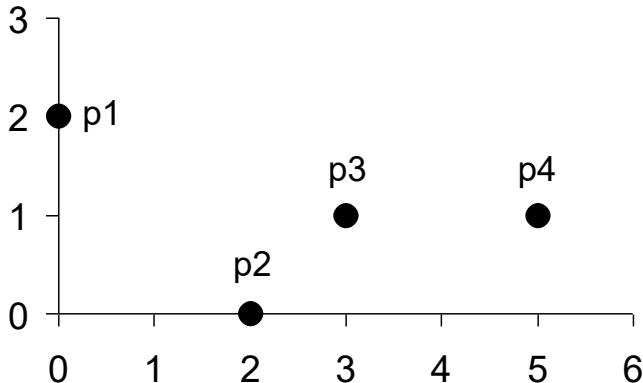
$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{1/r}$$

- where r is a parameter, n is the number of dimensions (attributes) and x_k and y_k are respectively, the k^{th} attributes (components) of data objects \mathbf{x} and \mathbf{y} .

Distances - Minkowski Distance

- The following are the three most common examples of Minkowski distances.
 - $r = 1$, City block (Manhattan, taxicab, L_1 norm) distance.
 - A common example of this for binary vectors is the Hamming distance, which is just the number of bits that are different between two binary vectors
 - $r = 2$, Euclidean distance (L_2 norm)
 - $r = \infty$, Supremum (L_{\max} norm, L_∞ norm) distance.
 - This is the maximum difference between any component of the vectors
- Do not confuse r with n , i.e., all these distances are defined for all numbers of dimensions.

Distances - Minkowski Distance



point	x	y
p1	0	2
p2	2	0
p3	3	1
p4	5	1

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^n |x_k - y_k|^r \right)^{1/r}$$

L1	p1	p2	p3	p4
p1	0	4	4	6
p2	4	0	2	4
p3	4	2	0	2
p4	6	4	2	0

L2	p1	p2	p3	p4
p1	0	2.828	3.162	5.099
p2	2.828	0	1.414	3.162
p3	3.162	1.414	0	2
p4	5.099	3.162	2	0

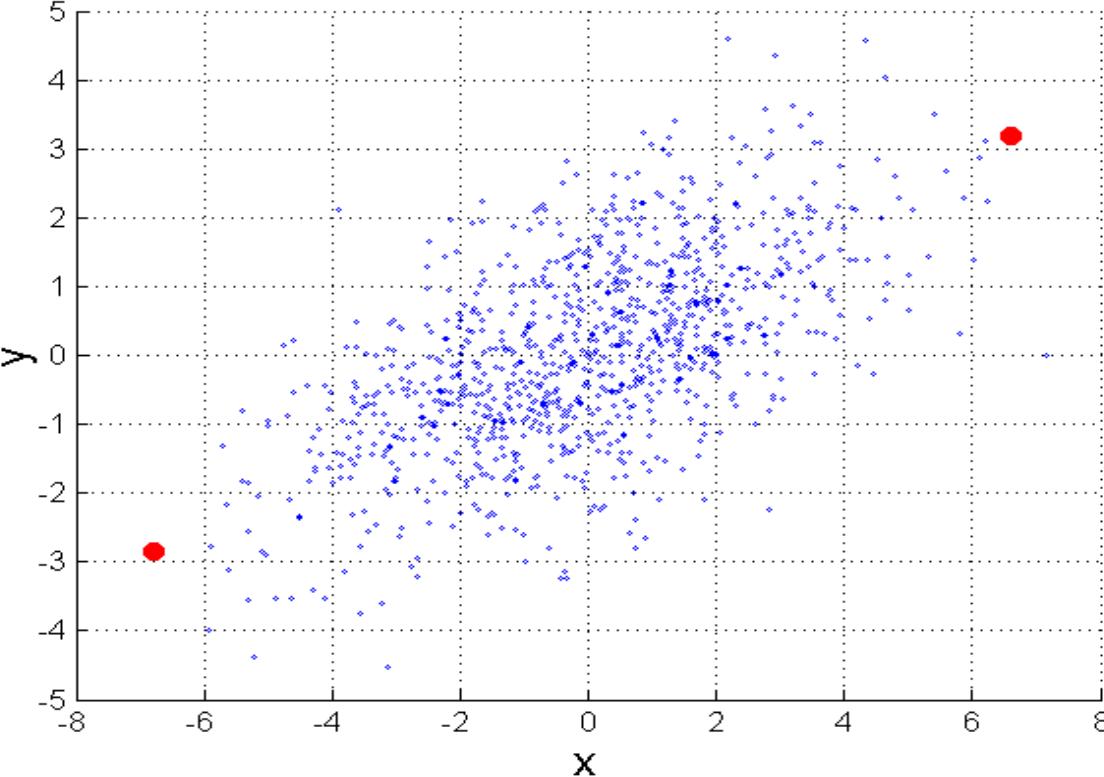
L ∞	p1	p2	p3	p4
p1	0	2	3	5
p2	2	0	1	3
p3	3	1	0	2
p4	5	3	2	0

Distance Matrix

Distances - Mahalanobis Distance

- Mahalonobis distance is the distance between a point and a distribution (not between two distinct points).
 - It is effectively a multivariate equivalent of the Euclidean distance.
 - It transforms the columns into uncorrelated variables
 - Scale the columns to make their variance equal to 1
 - Finally, it calculates the Euclidean distance.
- It is defined as
$$\text{Mahalanobis}(\mathbf{x}, \mathbf{y}) = \sqrt{(\mathbf{x} - \mathbf{y})' \Sigma^{-1} (\mathbf{x} - \mathbf{y})}$$
 - where Σ^{-1} is the inverse of the covariance matrix of the data.

Distances - Mahalanobis Distance

- In the Figure, there are 1000 points, whose x and y attributes have a correlation of 0.6.
 - The Euclidean distance between the two large points at the opposite ends of the long axis of the ellipse is 14.7, but Mahalanobis distance is only 6.
 - This is because the Mahalanobis distance gives less emphasis to the direction of largest variance.
- 
- A scatter plot showing 1000 data points. The x-axis ranges from -8 to 8, and the y-axis ranges from -5 to 5. The data points form an elliptical cluster centered at the origin (0,0). The horizontal axis of the ellipse is approximately 6 units long, and the vertical axis is approximately 3 units long. Two points are highlighted in red: one at approximately (-7, -3) and another at approximately (7, 3), which are at the opposite ends of the major axis. The plot includes a grid with dashed lines.

Distances - Mahalanobis Distance

- Covariance Matrix:

$$\Sigma = \begin{bmatrix} 0.3 & 0.2 \\ 0.2 & 0.3 \end{bmatrix}$$

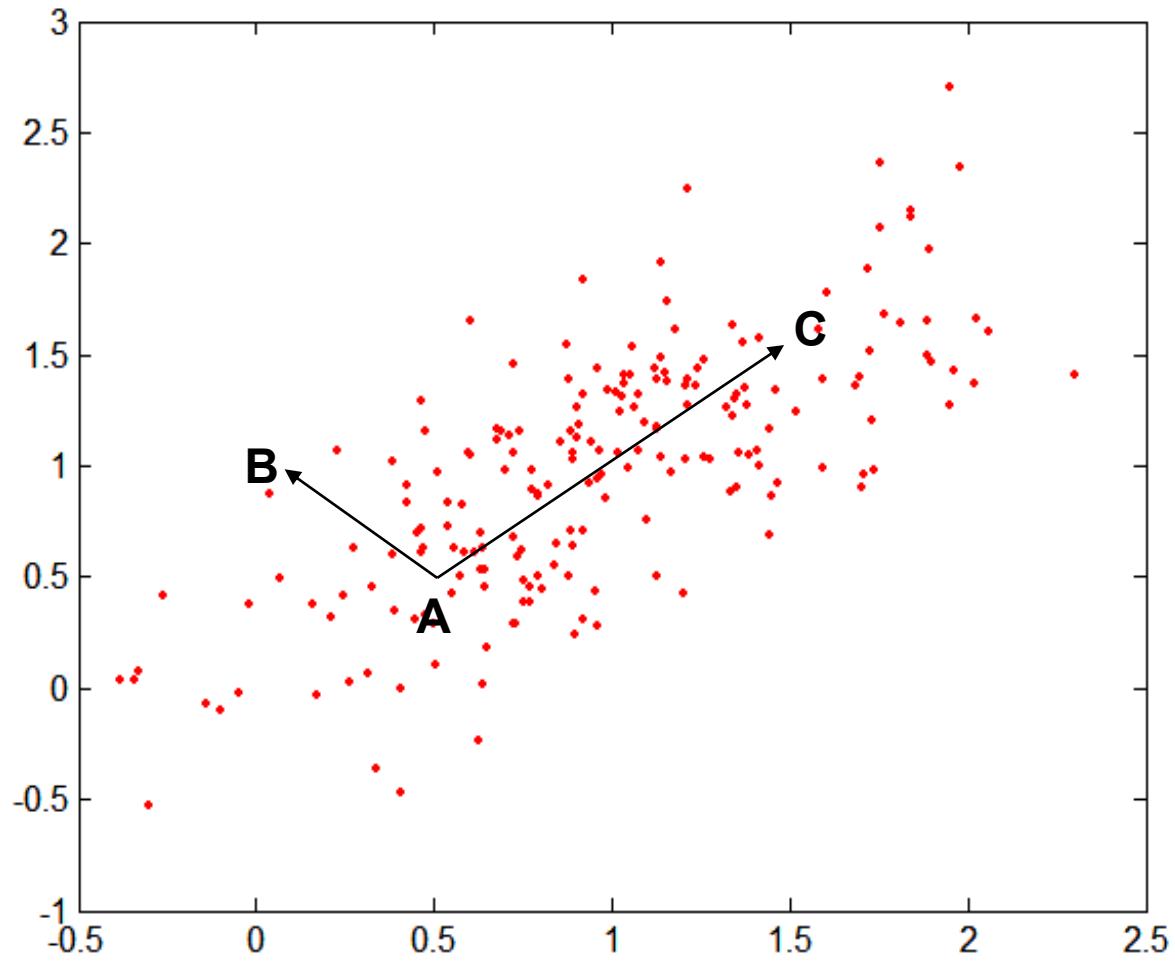
A: (0.5, 0.5)

B: (0, 1)

C: (1.5, 1.5)

$\text{Mahal}(A, B) = 5$

$\text{Mahal}(A, C) = 4$



Common Properties of a Distance

- Distances, such as the Euclidean distance, have some well-known properties.
- If $d(x, y)$ is the distance between two points, x and y , then the following properties hold.
 - Positivity
 - $d(x, y) \geq 0$ for all x and y
 - $d(x, y) = 0$ only if $x = y$
 - Symmetry
 - $d(x, y) = d(y, x)$ for all x and y
 - Triangle Inequality
 - $d(x, z) \leq d(x, y) + d(y, z)$ for all points x , y , and z
- Measures that satisfy all three properties are known as **metrics**

Common Properties of a Similarity

- If $s(x, y)$ is the similarity between points x and y , then the typical properties of similarities are the following:
 - Positivity
 - $s(x, y) = 1$ only if $x = y$. ($0 \leq s \leq 1$)
 - Symmetry
 - $s(x, y) = s(y, x)$ for all x and y
- For similarities, the triangle inequality typically does not hold
 - However, a similarity measure can be converted to a metric distance

A Non-symmetric Similarity Measure Example

- Consider an experiment in which people are asked to classify a small set of characters as they flash on a screen.
 - The confusion matrix for this experiment records how often each character is classified as itself, and how often each is classified as another character.
 - Using the confusion matrix, we can define a similarity measure between a character x and a character y as the number of times that x is misclassified as y ,
 - but note that this measure is not symmetric.

A Non-symmetric Similarity Measure Example

- For example, suppose that “0” appeared 200 times and was classified as a “0” 160 times, but as an “o” 40 times.
- Likewise, suppose that “o” appeared 200 times and was classified as an “o” 170 times, but as “0” only 30 times.
 - Then, $s(0,o) = 40$, but $s(o, 0) = 30$.
- In such situations, the similarity measure can be made symmetric by setting
 - $s'(x, y) = s'(y, x) = (s(x, y) + s(y, x))/2$,
 - where s indicates the new similarity measure.

Similarity Measures for Binary Data

- Similarity measures between objects that contain only binary attributes are called **similarity coefficients**, and typically have values between 0 and 1.
- Let x and y be two objects that consist of n binary attributes.
 - The comparison of two binary vectors, leads to the following quantities (frequencies):
 - f_{00} = the number of attributes where x is 0 and y is 0
 - f_{01} = the number of attributes where x is 0 and y is 1
 - f_{10} = the number of attributes where x is 1 and y is 0
 - f_{11} = the number of attributes where x is 1 and y is 1

Similarity Measures for Binary Data

- Simple Matching Coefficient (SMC)
 - One commonly used similarity coefficient

$$SMC = \frac{\text{number of matching attribute values}}{\text{number of attributes}} = \frac{f_{11} + f_{00}}{f_{01} + f_{10} + f_{11} + f_{00}}$$

- This measure counts both presences and absences equally.
 - Consequently, the SMC could be used to find students who had answered questions similarly on a test that consisted only of true/false questions.

Similarity Measures for Binary Data

- Jaccard Similarity Coefficient
 - frequently used to handle objects consisting of asymmetric binary attributes
 - This measure counts both presences and absences equally.
 - Consequently, the SMC could be used to find students who had answered questions similarly on a test that consisted only of true/false questions.

$$J = \frac{\text{number of matching presences}}{\text{number of attributes not involved in 00 matches}} = \frac{f_{11}}{f_{01} + f_{10} + f_{11}}$$

SMC versus Jaccard: Example

- Calculate SMC and J for the binary vectors,

$$x = (1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0)$$

$$y = (0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 0 \ 0 \ 1)$$

$f_{01} = 2$ (the number of attributes where x was 0 and y was 1)

$f_{10} = 1$ (the number of attributes where x was 1 and y was 0)

$f_{00} = 7$ (the number of attributes where x was 0 and y was 0)

$f_{11} = 0$ (the number of attributes where x was 1 and y was 1)

-

$$\begin{aligned} \text{SMC} &= (f_{11} + f_{00}) / (f_{01} + f_{10} + f_{11} + f_{00}) \\ &= (0 + 7) / (2 + 1 + 0 + 7) = 0.7 \end{aligned}$$

$$\begin{aligned} J &= (f_{11}) / (f_{01} + f_{10} + f_{11}) \\ &= 0 / (2 + 1 + 0) = 0 \end{aligned}$$

Cosine Similarity

- Cosine Similarity is one of the most common measures of document similarity
- If \mathbf{x} and \mathbf{y} are two document vectors, then

$$\cos(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{\mathbf{x}' \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|}$$

— where ' $'$ indicates vector or matrix transpose and $\langle \mathbf{x}, \mathbf{y} \rangle$ indicates the inner product of the two vectors,

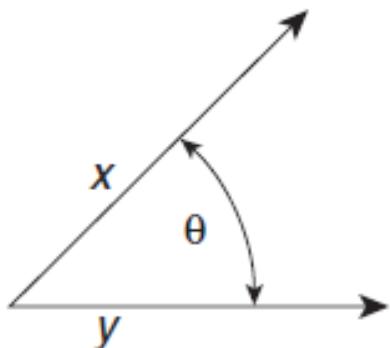
$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{k=1}^n x_k y_k = \mathbf{x}' \mathbf{y} \text{ and } \|\mathbf{x}\| \text{ is the length of vector } \mathbf{x},$$

$$\|\mathbf{x}\| = \sqrt{\sum_{k=1}^n x_k^2} = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\mathbf{x}' \mathbf{x}}$$

Cosine Similarity

- Cosine similarity really is a measure of the (cosine of the) angle between x and y .
 - Thus, if the cosine similarity is 1, the angle between x and y is 0° , and x and y are the same except for length.
 - If the cosine similarity is 0, then the angle between x and y is 90° , and they do not share any terms (words).
- It can also be written as

$$\cos(x, y) = \left\langle \frac{x}{\|x\|}, \frac{y}{\|y\|} \right\rangle = \langle x', y' \rangle$$



Cosine Similarity - Example

- Cosine Similarity between two document vectors
- This example calculates the cosine similarity for the following two data objects, which might represent document vectors:

$$x = (3, 2, 0, 5, 0, 0, 0, 2, 0, 0)$$

$$y = (1, 0, 0, 0, 0, 0, 0, 1, 0, 2)$$

$$\langle x, y \rangle = 3 \times 1 + 2 \times 0 + 0 \times 0 + 5 \times 0 + 0 \times 0 + 0 \times 0 + \\ 0 \times 0 + 2 \times 1 + 0 \times 0 + 0 \times 2 = 5$$

$$\|x\| = \sqrt{3^2 + 2^2 + 0^2 + 5^2 + 0^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2} = 6.48$$

$$\|y\| = \sqrt{1^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 0^2 + 1^2 + 0^2 + 2^2} = 2.45$$

$$\cos(x, y) = \frac{\langle x, y \rangle}{\|x\| \times \|y\|} = \frac{5}{6.48 \times 2.45} = 0.31$$

Extended Jaccard Coefficient

- Also known as Tanimoto Coefficient
- The extended Jaccard coefficient can be used for document data and that reduces to the Jaccard coefficient in the case of binary attributes.
- This coefficient, which we shall represent as EJ, is defined by the following equation:

$$EJ(\mathbf{x}, \mathbf{y}) = \frac{\langle \mathbf{x}, \mathbf{y} \rangle}{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \langle \mathbf{x}, \mathbf{y} \rangle} = \frac{\mathbf{x}'\mathbf{y}}{\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 - \mathbf{x}'\mathbf{y}}$$

Correlation

- used to measure the linear relationship between two sets of values that are observed together.
 - Thus, correlation can measure the relationship between two variables (height and weight) or between two objects (a pair of temperature time series).
- Correlation is used much more frequently to measure the similarity between attributes
 - since the values in two data objects come from different attributes, which can have very different attribute types and scales.
- There are many types of correlation

Correlation - Pearson's correlation

- between two sets of numerical values, i.e., two vectors, \mathbf{x} and \mathbf{y} , is defined by:

$$\text{corr}(\mathbf{x}, \mathbf{y}) = \frac{\text{covariance}(\mathbf{x}, \mathbf{y})}{\text{standard_deviation}(\mathbf{x}) \times \text{standard_deviation}(\mathbf{y})} = \frac{s_{xy}}{s_x s_y}$$

- where the following standard statistical notation and definitions are used:

$$\text{covariance}(\mathbf{x}, \mathbf{y}) = s_{xy} = \frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})(y_k - \bar{y})$$

$$\text{standard_deviation}(\mathbf{x}) = s_x = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (x_k - \bar{x})^2}$$

$$\text{standard_deviation}(\mathbf{y}) = s_y = \sqrt{\frac{1}{n-1} \sum_{k=1}^n (y_k - \bar{y})^2}$$

$$\bar{x} = \frac{1}{n} \sum_{k=1}^n x_k \text{ is the mean of } \mathbf{x}$$

$$\bar{y} = \frac{1}{n} \sum_{k=1}^n y_k \text{ is the mean of } \mathbf{y}$$

Correlation – Example (Perfect Correlation)

- Correlation is always in the range -1 to 1 .
 - A correlation of 1 (-1) means that x and y have a perfect positive (negative) linear relationship;
 - that is, $x_k = ay_k + b$, where a and b are constants.
- The following two vectors x and y illustrate cases where the correlation is -1 and $+1$, respectively.

$$x = (-3, 6, 0, 3, -6)$$

$$y = (1, -2, 0, -1, 2)$$

$$x = (3, 6, 0, 3, 6)$$

$$y = (1, 2, 0, 1, 2)$$

$$\text{corr}(x, y) = -1 \quad x_k = -3y_k \quad \text{corr}(x, y) = 1 \quad x_k = 3y_k$$

Correlation – Example (Nonlinear Relationships)

- If the correlation is 0, then there is no linear relationship between the two sets of values.
 - However, nonlinear relationships can still exist.
 - In the following example, $y_k = x_k^2$, but their correlation is 0.

$$x = (-3, -2, -1, 0, 1, 2, 3)$$

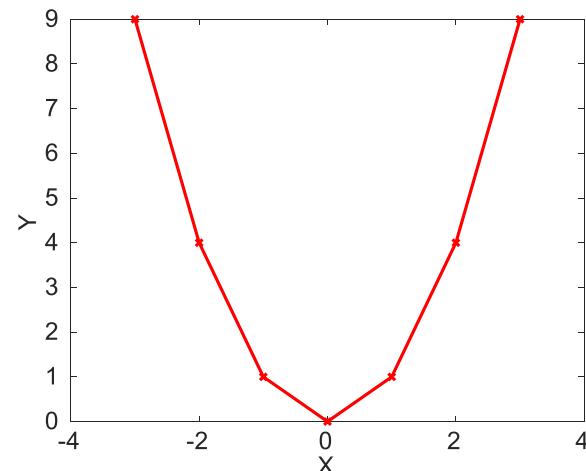
$$y = (9, 4, 1, 0, 1, 4, 9)$$

$$y_k = x_k^2$$

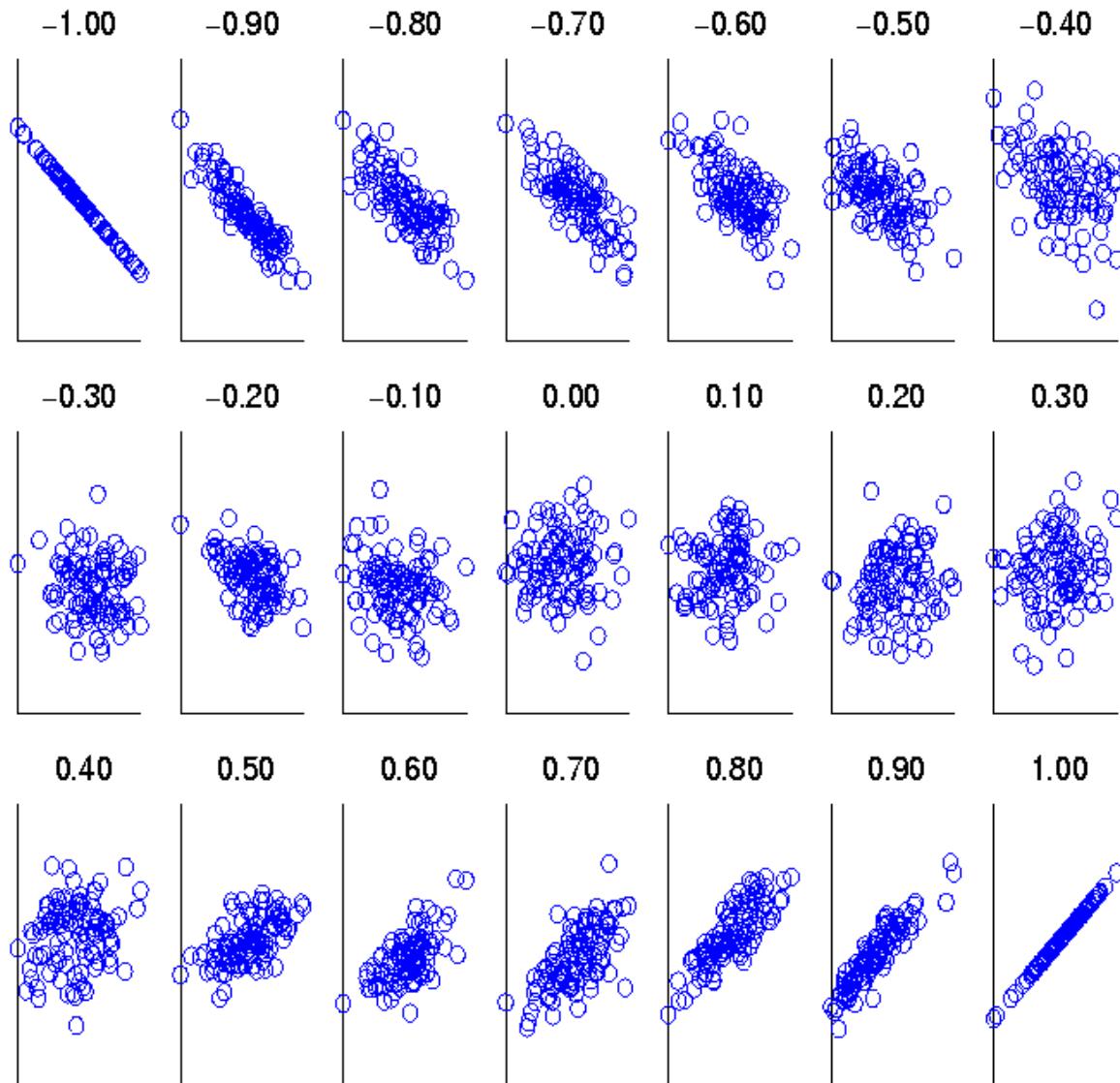
$$\text{mean}(x) = 0, \text{mean}(y) = 4$$

$$\text{std}(x) = 2.16, \text{std}(y) = 3.74$$

$$\text{corr} = \frac{(-3)(5)+(-2)(0)+(-1)(-3)+(0)(-4)+(1)(-3)+(2)(0)+(3)(5)}{6 \times 2.16 \times 3.74} = 0$$



Visually Evaluating Correlation



- Scatter plots showing the similarity from -1 to 1 .

Correlation vs Cosine vs Euclidean Distance

- Compare the three proximity measures according to their behavior under variable transformation
 - scaling: multiplication by a value
 - translation: adding a constant

Property	Cosine	Correlation	Euclidean Distance
Invariant to scaling (multiplication)	Yes	Yes	No
Invariant to translation (addition)	No	Yes	No

- Consider the example
 - $x = (1, 2, 4, 3, 0, 0, 0)$, $y = (1, 2, 3, 4, 0, 0, 0)$
 - $y_s = y \times 2 = (2, 4, 6, 8, 0, 0, 0)$ $y_t = y + 5 = (6, 7, 8, 9, 5, 5, 5)$

Measure	(x, y)	(x, y_s)	(x, y_t)
Cosine	0.9667	0.9667	0.7940
Correlation	0.9429	0.9429	0.9429
Euclidean Distance	1.4142	5.8310	14.2127

Correlation vs cosine vs Euclidean distance

- Choice of the right proximity measure depends on the domain
- What is the correct choice of proximity measure for the following situations?
 - Comparing documents using the frequencies of words
 - Documents are considered similar if the word frequencies are similar
 - Comparing the temperature in Celsius of two locations
 - Two locations are considered similar if the temperatures are similar in magnitude
 - Comparing two time series of temperature measured in Celsius
 - Two time series are considered similar if their shape is similar,
 - i.e., they vary in the same way over time, achieving minimums and maximums at similar times, etc.

Comparison of Proximity Measures

- Domain of application
 - Similarity measures tend to be specific to the type of attribute and data
 - Record data, images, graphs, sequences, 3D-protein structure, etc. tend to have different measures
- However, one can talk about various properties that you would like a proximity measure to have
 - Symmetry is a common one
 - Tolerance to noise and outliers is another
 - Ability to find more types of patterns?
 - Many others possible
- The measure must be applicable to the data and produce results that agree with domain knowledge

Information Based Measures

- Information theory is a well-developed and fundamental discipline with broad applications
- Some similarity measures are based on information theory
 - Mutual information in various versions
 - Maximal Information Coefficient (MIC) and related measures
 - General and can handle non-linear relationships
 - Can be complicated and time intensive to compute

- Information relates to possible outcomes of an event
 - transmission of a message, flip of a coin, or measurement of a piece of data



- The more certain an outcome, the less information that it contains and vice-versa
 - For example, if a coin has two heads, then an outcome of heads provides no information
 - More quantitatively, the information is related the probability of an outcome
 - The smaller the probability of an outcome, the more information it provides and vice-versa
 - Entropy is the commonly used measure

Entropy

- For
 - a variable (event), X ,
 - with n possible values (outcomes), $x_1, x_2 \dots, x_n$
 - each outcome having probability, $p_1, p_2 \dots, p_n$
 - the entropy of X , $H(X)$, is given by

$$H(X) = - \sum_{i=1}^n p_i \log_2 p_i$$

- Entropy is between 0 and $\log_2 n$ and is measured in bits
 - Thus, entropy is a measure of how many bits it takes to represent an observation of X on average

Entropy Examples

- For a coin with probability p of heads and probability $q = 1 - p$ of tails

$$H = -p \log_2 p - q \log_2 q$$

- For $p=0.5, q=0.5$ (fair coin) $H=1$
- For $p = 1$ or $q = 1$, $H = 0$
- What is the entropy of a fair four-sided die?

Entropy for Sample Data: Example

Hair Color	Count	p	$-p \log_2 p$
Black	75	0.75	0.3113
Brown	15	0.15	0.4105
Blond	5	0.05	0.2161
Red	0	0.00	0
Other	5	0.05	0.2161
Total	100	1.0	1.1540

- Maximum entropy is $\log_2 5 = 2.3219$

Entropy for Sample Data

- Suppose we have
 - a number of observations (m) of some attribute, X , e.g., the hair color of students in the class,
 - where there are n different possible values
 - And the number of observation in the i^{th} category is m_i
 - Then, for this sample

$$H(X) = - \sum_{i=1}^n \frac{m_i}{m} \log_2 \frac{m_i}{m}$$

- For continuous data, the calculation is harder

Mutual Information

- used as a measure of similarity between two sets of paired values that is sometimes used as an alternative to correlation, particularly when a nonlinear relationship is suspected between the pairs of values.
 - This measure comes from information theory, which is the study of how to formally define and quantify information.
 - It is a measure of how much information one set of values provides about another, given that the values come in pairs, e.g., height and weight.
 - If the two sets of values are independent, i.e., the value of one tells us nothing about the other, then their mutual information is 0.

Mutual Information

- Information one variable provides about another

Formally, $I(X, Y) = H(X) + H(Y) - H(X, Y)$, where $H(X, Y)$ is the joint entropy of X and Y ,

$$H(X, Y) = - \sum_i \sum_j p_{ij} \log_2 p_{ij}$$

where p_{ij} is the probability that the i^{th} value of X and the j^{th} value of Y occur together

- For discrete variables, this is easy to compute
- Maximum mutual information for discrete variables is $\log_2(\min(n_X, n_Y))$, where $n_X (n_Y)$ is the number of values of $X (Y)$

Mutual Information Example

- Evaluating Nonlinear Relationships with Mutual Information
 - Recall Example where $y_k = x_k^2$, but their correlation was 0.

$$x = (-3, -2, -1, 0, 1, 2, 3)$$

$$y = (9, 4, 1, 0, 1, 4, 9)$$

$$I(x, y) = H(x) + H(y) - H(x, y) = 1.9502$$

Entropy for y

x_j	$P(x = x_j)$	$-P(x = x_j) \log_2 P(x = x_j)$
-3	1/7	0.4011
-2	1/7	0.4011
-1	1/7	0.4011
0	1/7	0.4011
1	1/7	0.4011
2	1/7	0.4011
3	1/7	0.4011
$H(x)$		2.8074

Entropy for x

y_k	$P(y = y_k)$	$-P(y = y_k) \log_2 P(y = y_k)$
9	2/7	0.5164
4	2/7	0.5164
1	2/7	0.5164
0	1/7	0.4011
$H(y)$		1.9502

Joint entropy for x and y

x_j	y_k	$P(x = x_j, y = y_k)$	$-P(x = x_j, y = y_k) \log_2 P(x = x_j, y = y_k)$
-3	9	1/7	0.4011
-2	4	1/7	0.4011
-1	1	1/7	0.4011
0	0	1/7	0.4011
1	1	1/7	0.4011
2	4	1/7	0.4011
3	9	1/7	0.4011
$H(x, y)$			2.8074

Mutual Information Example

Student Status	Count	p	$-p \log_2 p$
Undergrad	45	0.45	0.5184
Grad	55	0.55	0.4744
Total	100	1.00	0.9928

Grade	Count	p	$-p \log_2 p$
A	35	0.35	0.5301
B	50	0.50	0.5000
C	15	0.15	0.4105
Total	100	1.00	1.4406

Student Status	Grade	Count	p	$-p \log_2 p$
Undergrad	A	5	0.05	0.2161
Undergrad	B	30	0.30	0.5211
Undergrad	C	10	0.10	0.3322
Grad	A	30	0.30	0.5211
Grad	B	20	0.20	0.4644
Grad	C	5	0.05	0.2161
Total		100	1.00	2.2710

- Mutual information of Student Status and Grade
 $= 0.9928 + 1.4406 - 2.2710 = 0.1624$

Maximal Information Coefficient

- Applies mutual information to two continuous variables
- Consider the possible binnings of the variables into discrete categories
 - $n_X \times n_Y \leq N^{0.6}$ where
 - n_X is the number of values of X
 - n_Y is the number of values of Y
 - N is the number of samples (observations, data objects)
- Compute the mutual information
 - Normalized by $\log_2(\min(n_X, n_Y))$
- Take the highest value
- Reshef, David N., Yakir A. Reshef, Hilary K. Finucane, Sharon R. Grossman, Gilean McVean, Peter J. Turnbaugh, Eric S. Lander, Michael Mitzenmacher, and Pardis C. Sabeti. "Detecting novel associations in large data sets." *science* 334, no. 6062 (2011): 1518-1524.

General Approach for Combining Similarities

- Sometimes attributes are of many different types, but an overall similarity is needed.
 - For the k^{th} attribute, compute a similarity, $s_k(x, y)$, in the range $[0, 1]$.
 - Define an indicator variable, δ_k , for the k^{th} attribute as follows:
 - $\delta_k = 0$ if the k^{th} attribute is an asymmetric attribute and both objects have a value of 0, or if one of the objects has a missing value for the k^{th} attribute
 - $\delta_k = 1$ otherwise
 - Compute

$$\text{similarity}(x, y) = \frac{\sum_{k=1}^n \delta_k s_k(x, y)}{\sum_{k=1}^n \delta_k}$$

Using Weights to Combine Similarities

- May not want to treat all attributes the same.
 - Use non-negative weights ω_k

- $similarity(\mathbf{x}, \mathbf{y}) = \frac{\sum_{k=1}^n \omega_k \delta_k s_k(\mathbf{x}, \mathbf{y})}{\sum_{k=1}^n \omega_k \delta_k}$

- Can also define a weighted form of distance

$$d(\mathbf{x}, \mathbf{y}) = \left(\sum_{k=1}^n w_k |x_k - y_k|^r \right)^{1/r}$$

Data Mining: Concepts and Techniques

— Chapter 2 —

Jiawei Han, Micheline Kamber, and Jian Pei

University of Illinois at Urbana-Champaign

Simon Fraser University

©2011 Han, Kamber, and Pei. All rights reserved.

Chapter 2: Getting to Know Your Data

- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization 
- Measuring Data Similarity and Dissimilarity
- Summary

Data Visualization

- Why data visualization?
 - Gain insight into an information space by mapping data onto graphical primitives
 - Provide qualitative overview of large data sets
 - Search for patterns, trends, structure, irregularities, relationships among data
 - Help find interesting regions and suitable parameters for further quantitative analysis
 - Provide a visual proof of computer representations derived
- Categorization of visualization methods:
 - Pixel-oriented visualization techniques
 - Geometric projection visualization techniques
 - Icon-based visualization techniques
 - Hierarchical visualization techniques
 - Visualizing complex data and relations

Chapter 2: Getting to Know Your Data

- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization
- Measuring Data Similarity and Dissimilarity
- Summary



Similarity and Dissimilarity

- **Similarity**

- Numerical measure of how alike two data objects are
- Value is higher when objects are more alike
- Often falls in the range [0,1]

- **Dissimilarity** (e.g., distance)

- Numerical measure of how different two data objects are
- Lower when objects are more alike
- Minimum dissimilarity is often 0
- Upper limit varies

- **Proximity** refers to a similarity or dissimilarity

Data Matrix and Dissimilarity Matrix

■ Data matrix

- n data points with p dimensions
- Two modes

$$\begin{bmatrix} x_{11} & \dots & x_{1f} & \dots & x_{1p} \\ \dots & \dots & \dots & \dots & \dots \\ x_{i1} & \dots & x_{if} & \dots & x_{ip} \\ \dots & \dots & \dots & \dots & \dots \\ x_{n1} & \dots & x_{nf} & \dots & x_{np} \end{bmatrix}$$

■ Dissimilarity matrix

- n data points, but registers only the distance
- A triangular matrix
- Single mode

$$\begin{bmatrix} 0 & & & & \\ d(2,1) & 0 & & & \\ d(3,1) & d(3,2) & 0 & & \\ \vdots & \vdots & \vdots & \ddots & \\ d(n,1) & d(n,2) & \dots & \dots & 0 \end{bmatrix}$$

Proximity Measure for Nominal Attributes

- Can take 2 or more states, e.g., red, yellow, blue, green (generalization of a binary attribute)
- Method 1: Simple matching
 - m : # of matches, p : total # of variables

$$d(i, j) = \frac{p - m}{p}$$

- Method 2: Use a large number of binary attributes
 - creating a new binary attribute for each of the M nominal states

Proximity Measure for Binary Attributes

- A contingency table for binary data

		Object <i>j</i>		
		1	0	sum
Object <i>i</i>	1	<i>q</i>	<i>r</i>	<i>q + r</i>
	0	<i>s</i>	<i>t</i>	<i>s + t</i>
sum		<i>q + s</i>	<i>r + t</i>	<i>p</i>

- Distance measure for symmetric binary variables:

$$d(i, j) = \frac{r + s}{q + r + s + t}$$

- Distance measure for asymmetric binary variables:

$$d(i, j) = \frac{r + s}{q + r + s}$$

- Jaccard coefficient (*similarity* measure for *asymmetric* binary variables):

$$\text{sim}_{\text{Jaccard}}(i, j) = \frac{q}{q + r + s}$$

- Note: Jaccard coefficient is the same as "coherence":

$$\text{coherence}(i, j) = \frac{\text{sup}(i, j)}{\text{sup}(i) + \text{sup}(j) - \text{sup}(i, j)} = \frac{q}{(q + r) + (q + s) - q}$$

Dissimilarity between Binary Variables

■ Example

Name	Gender	Fever	Cough	Test-1	Test-2	Test-3	Test-4
Jack	M	Y	N	P	N	N	N
Mary	F	Y	N	P	N	P	N
Jim	M	Y	P	N	N	N	N

- Gender is a symmetric attribute
- The remaining attributes are asymmetric binary
- Let the values Y and P be 1, and the value N 0

$$d(jack, mary) = \frac{0 + 1}{2 + 0 + 1} = 0.33$$

$$d(jack, jim) = \frac{1 + 1}{1 + 1 + 1} = 0.67$$

$$d(jim, mary) = \frac{1 + 2}{1 + 1 + 2} = 0.75$$

Standardizing Numeric Data

- Z-score:
$$z = \frac{x - \mu}{\sigma}$$
 - X: raw score to be standardized, μ : mean of the population, σ : standard deviation
 - the distance between the raw score and the population mean in units of the standard deviation
 - negative when the raw score is below the mean, "+" when above
- An alternative way: Calculate the mean absolute deviation

$$s_f = \frac{1}{n}(|x_{1f} - m_f| + |x_{2f} - m_f| + \dots + |x_{nf} - m_f|)$$

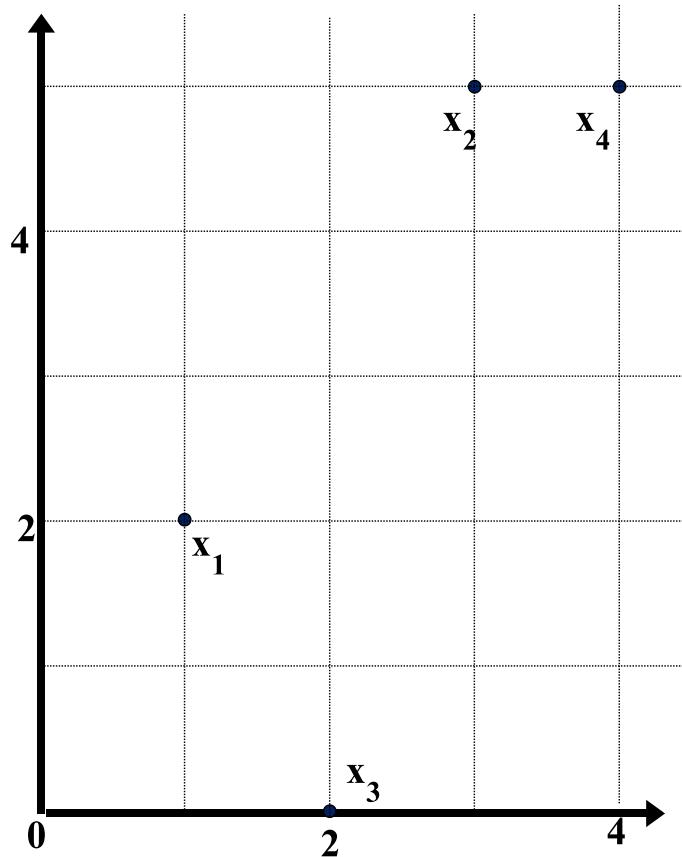
where $m_f = \frac{1}{n}(x_{1f} + x_{2f} + \dots + x_{nf})$.

$$z_{if} = \frac{x_{if} - m_f}{s_f}$$

- standardized measure (*z-score*):
- Using mean absolute deviation is more robust than using standard deviation

Example:

Data Matrix and Dissimilarity Matrix



Data Matrix

point	attribute1	attribute2
$x1$	1	2
$x2$	3	5
$x3$	2	0
$x4$	4	5

Dissimilarity Matrix

(with Euclidean Distance)

	$x1$	$x2$	$x3$	$x4$
$x1$	0			
$x2$	3.61	0		
$x3$	5.1	5.1	0	
$x4$	4.24	1	5.39	0

Distance on Numeric Data: Minkowski Distance

- *Minkowski distance*: A popular distance measure

$$d(i, j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

where $i = (x_{i1}, x_{i2}, \dots, x_{ip})$ and $j = (x_{j1}, x_{j2}, \dots, x_{jp})$ are two p -dimensional data objects, and h is the order (the distance so defined is also called L- h norm)

- Properties
 - $d(i, j) > 0$ if $i \neq j$, and $d(i, i) = 0$ (Positive definiteness)
 - $d(i, j) = d(j, i)$ (Symmetry)
 - $d(i, j) \leq d(i, k) + d(k, j)$ (Triangle Inequality)
- A distance that satisfies these properties is a **metric**

Special Cases of Minkowski Distance

- $h = 1$: Manhattan (city block, L_1 norm) distance
 - E.g., the Hamming distance: the number of bits that are different between two binary vectors

$$d(i, j) = |x_{i1} - x_{j1}| + |x_{i2} - x_{j2}| + \dots + |x_{ip} - x_{jp}|$$

- $h = 2$: (L_2 norm) Euclidean distance

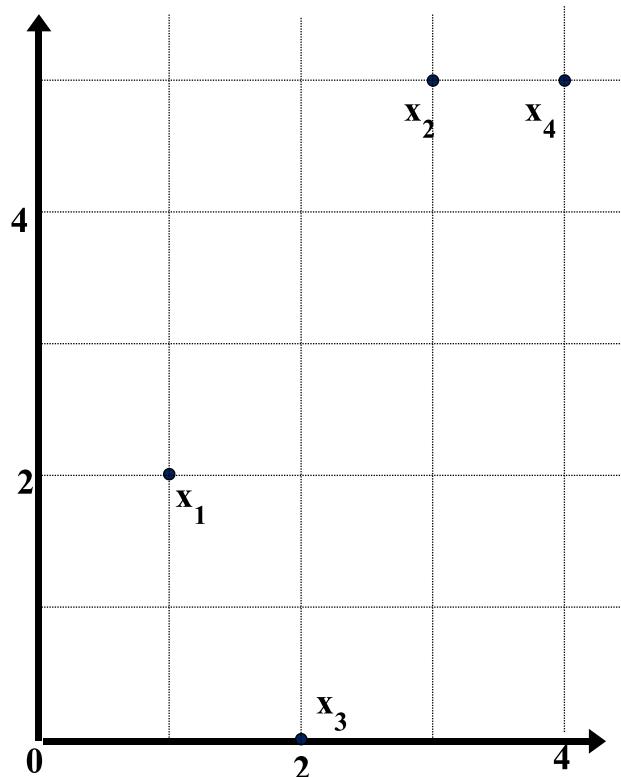
$$d(i, j) = \sqrt{(|x_{i1} - x_{j1}|^2 + |x_{i2} - x_{j2}|^2 + \dots + |x_{ip} - x_{jp}|^2)}$$

- $h \rightarrow \infty$. “supremum” (L_{\max} norm, L_∞ norm) distance.
 - This is the maximum difference between any component (attribute) of the vectors

$$d(i, j) = \lim_{h \rightarrow \infty} \left(\sum_{f=1}^p |x_{if} - x_{jf}|^h \right)^{\frac{1}{h}} = \max_f^p |x_{if} - x_{jf}|$$

Example: Minkowski Distance

point	attribute 1	attribute 2
x1	1	2
x2	3	5
x3	2	0
x4	4	5



Dissimilarity Matrices

Manhattan (L_1)

L	x1	x2	x3	x4
x1	0			
x2	5	0		
x3	3	6	0	
x4	6	1	7	0

Euclidean (L_2)

L2	x1	x2	x3	x4
x1	0			
x2	3.61	0		
x3	2.24	5.1	0	
x4	4.24	1	5.39	0

Supremum

L_∞	x1	x2	x3	x4
x1	0			
x2	3	0		
x3	2	5	0	
x4	3	1	5	0

Ordinal Variables

- An ordinal variable can be discrete or continuous
- Order is important, e.g., rank
- Can be treated like interval-scaled
 - replace x_{if} by their rank $r_{if} \in \{1, \dots, M_f\}$
 - map the range of each variable onto [0, 1] by replacing i -th object in the f -th variable by

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

- compute the dissimilarity using methods for interval-scaled variables

Attributes of Mixed Type

- A database may contain all attribute types
 - Nominal, symmetric binary, asymmetric binary, numeric, ordinal
- One may use a weighted formula to combine their effects

$$d(i, j) = \frac{\sum_{f=1}^p \delta_{ij}^{(f)} d_{ij}^{(f)}}{\sum_{f=1}^p \delta_{ij}^{(f)}}$$

- f is binary or nominal:
 $d_{ij}^{(f)} = 0$ if $x_{if} = x_{jf}$, or $d_{ij}^{(f)} = 1$ otherwise
- f is numeric: use the normalized distance
- f is ordinal
 - Compute ranks r_{if} and
 - Treat z_{if} as interval-scaled

$$z_{if} = \frac{r_{if} - 1}{M_f - 1}$$

Cosine Similarity

- A **document** can be represented by thousands of attributes, each recording the *frequency* of a particular word (such as keywords) or phrase in the document.

Document	team	coach	hockey	baseball	soccer	penalty	score	win	loss	season
Document1	5	0	3	0	2	0	0	2	0	0
Document2	3	0	2	0	1	1	0	1	0	1
Document3	0	7	0	2	1	0	0	3	0	0
Document4	0	1	0	0	1	2	2	0	3	0

- Other vector objects: gene features in micro-arrays, ...
- Applications: information retrieval, biologic taxonomy, gene feature mapping, ...
- Cosine measure: If d_1 and d_2 are two vectors (e.g., term-frequency vectors), then

$$\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|,$$

where \bullet indicates vector dot product, $\|d\|$: the length of vector d

Example: Cosine Similarity

- $\cos(d_1, d_2) = (d_1 \bullet d_2) / \|d_1\| \|d_2\|$,
where \bullet indicates vector dot product, $\|d\|$: the length of vector d
- Ex: Find the **similarity** between documents 1 and 2.

$$d_1 = (5, 0, 3, 0, 2, 0, 0, 2, 0, 0)$$

$$d_2 = (3, 0, 2, 0, 1, 1, 0, 1, 0, 1)$$

$$d_1 \bullet d_2 = 5*3+0*0+3*2+0*0+2*1+0*1+0*1+2*1+0*0+0*1 = 25$$

$$\|d_1\| = (5^2 + 0^2 + 3^2 + 0^2 + 2^2 + 0^2 + 0^2 + 2^2 + 0^2 + 0^2)^{0.5} = (42)^{0.5} = 6.481$$

$$\|d_2\| = (3^2 + 0^2 + 2^2 + 0^2 + 1^2 + 1^2 + 0^2 + 1^2 + 0^2 + 1^2)^{0.5} = (17)^{0.5} = 4.12$$

$$\cos(d_1, d_2) = 0.94$$

Chapter 2: Getting to Know Your Data

- Data Objects and Attribute Types
- Basic Statistical Descriptions of Data
- Data Visualization
- Measuring Data Similarity and Dissimilarity
- Summary



Summary

- Data attribute types: nominal, binary, ordinal, interval-scaled, ratio-scaled
- Many types of data sets, e.g., numerical, text, graph, Web, image.
- Gain insight into the data by:
 - Basic statistical data description: central tendency, dispersion, graphical displays
 - Data visualization: map data onto graphical primitives
 - Measure data similarity
- Above steps are the beginning of data preprocessing.
- Many methods have been developed but still an active area of research.

References

- W. Cleveland, Visualizing Data, Hobart Press, 1993
- T. Dasu and T. Johnson. Exploratory Data Mining and Data Cleaning. John Wiley, 2003
- U. Fayyad, G. Grinstein, and A. Wierse. Information Visualization in Data Mining and Knowledge Discovery, Morgan Kaufmann, 2001
- L. Kaufman and P. J. Rousseeuw. Finding Groups in Data: an Introduction to Cluster Analysis. John Wiley & Sons, 1990.
- H. V. Jagadish, et al., Special Issue on Data Reduction Techniques. Bulletin of the Tech. Committee on Data Eng., 20(4), Dec. 1997
- D. A. Keim. Information visualization and visual data mining, IEEE trans. on Visualization and Computer Graphics, 8(1), 2002
- D. Pyle. Data Preparation for Data Mining. Morgan Kaufmann, 1999
- S. Santini and R. Jain," Similarity measures", IEEE Trans. on Pattern Analysis and Machine Intelligence, 21(9), 1999
- E. R. Tufte. The Visual Display of Quantitative Information, 2nd ed., Graphics Press, 2001
- C. Yu , et al., Visual data mining of multimedia data for social and behavioral studies, Information Visualization, 8(1), 2009