IF5181 Pengenalan Pola

Evaluasi Tugas Klasifikasi

Masayu Leylia Khodra

Tugas 1

- Pilihlah dataset dari https://archive.ics.uci.edu/ml/datasets.php
- Tools: Jupyter Notebook dengan python. Gunakan scikit-learn untuk konstruksi model klasifikasi (training data)
- Skenario:
  - split dataset menjadi 90:10 (train:test)
  - Lakukanlah holdout validation dengan split training data menjadi 90:10 (train:validasi)
    - Gunakanlah validation data untuk memilih model klasifikasi terbaik dari berbagai algoritma pembelajaran dengan variasi nilai parameter berbeda.
  - Lakukanlah 5-fold cross validation untuk training data.
    - Pilih model klasifikasi terbaik dari berbagai algoritma pembelajaran dengan variasi nilai parameter berbeda.
  - Gunakanlah test data untuk mendapatkan kinerja dari setiap skenario.
- Lakukanlah analisis di proses validasi dan testing

Catatan Hasil Penilaian 1

- Pemilihan dataset:
  - Dataset dengan akurasi 1 (training maupun testing):
    - Kasus generated dataset dengan rumus tertentu. Jika label sudah bisa ditentukan dengan rumus atau aturannya, tidak perlu menggunakan machine learning untuk model klasifikasi.
  - Dataset yang terlalu sedikit datanya

Catatan Hasil Penilaian 1 (lanj)

- Split dataset: pastikan distribusi training data dan test data sama (proporsi setiap kelas sama).
- Preproses data bisa dilakukan untuk meningkatkan kinerja data, tetapi hati-hati melakukannya.
  - Categorical encoding: langsung, label encoding, one-hot encoding
- Hasil eksperimen sulit dibaca: perlu tabel rekapitulasi untuk setiap kelompok validasi.

Catatan Hasil Penilaian 1 (lanj)

- Setelah mendapatkan model terbaik dari holdout validation dan model terbaik dari k-fold cross validation, hasil yang didapatkan:
  - Algoritma learning sama: cukup 1 full-training model untuk tes
  - Algoritma learning berbeda: cek apakah full-training model dari k-fold cross validation lebih baik dari holdout validation untuk kinerja tes?

Catatan Hasil Penilaian 1 (lanj)

- Analisis hasil umumnya belum dilakukan
  - tampilkan confusion matrix
  - Identifikasi kasus misklasifikasi
  - Skenario perbaikan
- Perbaikan kinerja dapat dilakukan dengan:
  - Praproses tambahan
  - Analisis data agar lebih paham data
  - Seleksi dan ekstraksi fitur baru

Data Quality: Common Problem

- Common problem in data science: noisy, missing, inconsistent, duplicate data.
- Another problem:
  - imbalanced dataset
  - Outliers (extreme values).
    An outlier is a piece of data that is an abnormal distance from other points.

Noisy Data

Noise types: class (label) noise and attribute noise

- Class noise: contradictory examples, mislabeled examples
- Attribute noise: erroneous (at data entry, violation of known data constraints)

Attr1	Attr2	Class
0.25	red	positive
0.25	red	negative
1.02	green	positive
0.99	green	negative

contradictory

error at
data entry

mislabeled

https://sci2s.ugr.es/noisydata

Exercise 1: Find Noisy Data

ID	A1	A2	A3	A4	*C*
1	1	2	1	1	y1
2	2	1	1	1	y2
3	3	5	2	4	y1
4	2	2	2	2	y2
5	3	5	2	4	y2
6	6	7	5	8	y1
7	9	4	3	1	y1

https://pdfs.semanticscholar.org/ee88/ae92d19e208f15d3613016266bb13da8fc98.pdf

Exercise 2: Find Noisy Data

	A	B	C	D	E	F	G
1	**First name**	**Last name**	**January**	**February**	**March**	**Q1 Sales**	**Region**
2	Darrel	Alston	$ 11,896	$ 2,552	$ 11,350	$ 25,798	East
3	David	Terrell	$ 9,763	$ 1,749	$ 8,678	$ 20,190	South
4	Gwendolyn	Cameron	$ 9,421	$ 5,585	$ 10,423	$ 25,429	East
5	Katell	Hall	$ 3,291	$ 2,610	$ 13,692	$ 19,593	Norht
6	Honorato	Howard	$ 11,746	$ 6,756	$ 10,471	$ 28,973	West
7	Nehru	Rose	$ 8,603	$ 5,907	$ 1,682	$ 16,192	West
8	Upton	Shields	$ 10,955	$ 4,914	$ 11,539	$ 27,408	West
9	Germane	Holman	$ 11,561	$ 8,547	$ 8,433	$ 28,541	North
10	Elliott	Hall	$ 9,318	$ 5,857	$ 4,935	$ 20,110	North
11	Illana	Erickson	$ 3,709	$ 13,401	$ 3,431	$ 20,541	Wst
12	Lani	Spears	$ 5,620	$ 14,252	$ 8,894	$ 28,766	East
13	Clementine	Pope	$ 8,901	$ 10,143	$ 13,573	$ 32,617	South

https://edu.gcfglobal.org/en/excel-tips/a-trick-for-finding-inconsistent-data/1/

Missing values Data

Missing values

PassengerId	Survived	Pclass	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
1	0	3	male	22	1	0	A/5 21171	7.25		S
2	1	1	female	38	1	0	PC 17599	71.2833	C85	C
3	1	3	female	26	0	0	STON/O2. 3101282	7.925		S
4	1	1	female	35	1	0	113803	53.1	C123	S
5	0	3	male	35	0	0	373450	8.05		S
6	0	3	male		0	0	330877	8.4583		Q

https://freecontent.manning.com/pre-processing-data-for-modeling/

Inconsistent Data

- Inconsistent data contain discrepancies in name or code, or discrepancies between duplicate records (from multiple sources)

Age	BirthDate	...
18	30 June 2000	...
19	30 June 2000	...
...	...	...

ID	GPA	...
01	3.25	...
01	3.67	...
...	...	...

ID	Rating	...
1	A	...
2	B	...
3	1	...
4	3.5	...

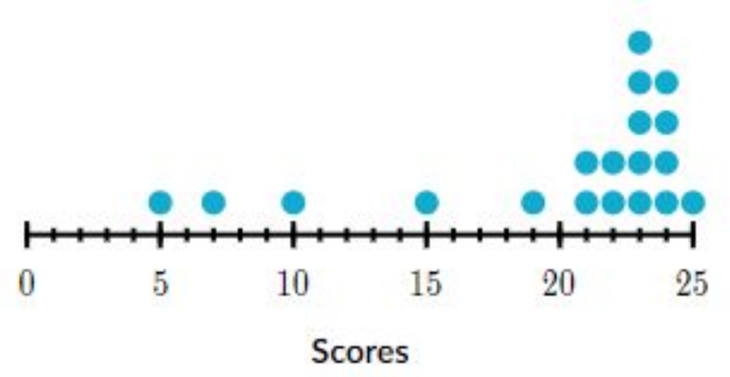
Imbalanced dataset

Class	Frequency
A	1000
B	10

Majority class

Minority class

Identify Outliers



19 Data:

5, 7, 10, 15, 19, 21, 21, 22, 22, 23, 23, 23, 23, 23, 24, 24, 24, 24, 25

Commonly used rules to identify outliers:

Low outlier < Q1-1.5*IQR

High outlier > Q3+1.5*IQR

Median: 23 ; Q1: 19 ; Q3: 24

IQR = Q3-Q1=24-19=5

Min = 19-7.5=11.5

Max = 24+7.5=31.5

https://www.khanacademy.org/math/statistics-probability/summarizing-quantitative-data/box-whisker-plots/a/identifying-outliers-iqr-rule

Exercise 3: Identify Outliers

	fixed acidity	volatile acidity	citric acid	residual sugar
count	4898.000000	4898.000000	4898.000000	4898.000000
mean	6.854788	0.278241	0.334192	6.391415
std	0.843868	0.100795	0.121020	5.072058
min	3.800000	0.080000	0.000000	0.600000
25%	6.300000	0.210000	0.270000	1.700000
50%	6.800000	0.260000	0.320000	5.200000
75%	7.300000	0.320000	0.390000	9.900000
max	14.200000	1.100000	1.660000	65.800000

Identify columns that have outliers ?

Verifying Data Quality

1. Identify incorrectness of data type assignment
2. Identify noise or inconsistent data
3. Identify missing values
4. Identify outliers

Incorrect data type assignment

```
In [15]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 205 entries, 0 to 204
Data columns (total 26 columns):
0      205 non-null int64
1      205 non-null object
2      205 non-null object
3      205 non-null object
4      205 non-null object
5      205 non-null object
6      205 non-null object
7      205 non-null object
8      205 non-null object
9      205 non-null float64
10     205 non-null float64
11     205 non-null float64
12     205 non-null float64
13     205 non-null int64
14     205 non-null object
15     205 non-null object
16     205 non-null int64
17     205 non-null object
18     205 non-null object
19     205 non-null object
20     205 non-null float64
21     205 non-null object
22     205 non-null object
23     205 non-null int64
24     205 non-null int64
25     205 non-null object
dtypes: float64(5), int64(5), object(16)
memory usage: 41.7+ KB
```

0. symboling: -3, -2, -1, 0, 1, 2, 3.
1. normalized-losses: continuous from 65 to 256.
2. make:
alfa-romero, audi, bmw, chevrolet, dodge, honda, isuzu, jaguar, mazda, mercedes-benz, mercury, mitsubishi, nissan, peugot, plymouth, porsche, renault, saab, subaru, toyota, volkswagen, volvo
3. fuel-type: diesel, gas.
4. aspiration: std, turbo.
5. num-of-doors: four, two.
6. body-style: hardtop, wagon, sedan, hatchback, convertibl
7. drive-wheels: 4wd, fwd, rwd.
8. engine-location: front, rear.
9. wheel-base: continuous from 86.6 120.9.
10. length: continuous from 141.1 to 208.1.
11. width: continuous from 60.3 to 72.3.
12. height: continuous from 47.8 to 59.8.
13. curb-weight: continuous from 1488 to 4066.
14. engine-type: dohc, dohcv, l, ohc, ohcf, ohcv, rotor.
15. num-of-cylinders: eight, five, four, six, three, twelve, two.
16. engine-size: continuous from 61 to 326.
17. fuel-system: 1bbl, 2bbl, 4bbl, idi, mfi, mpfi, spdi, spfi.
18. bore: continuous from 2.54 to 3.94.
19. stroke: continuous from 2.07 to 4.17.
20. compression-ratio: continuous from 7 to 23.
21. horsepower: continuous from 48 to 288.
22. peak-rpm: continuous from 4150 to 6600.
23. city-mpg: continuous from 13 to 49.
24. highway-mpg: continuous from 16 to 54.
25. price: continuous from 5118 to 45400.

Attribute: Identify Noise or Inconsistent Data

```
In [44]: df[23].value_counts()
Out[44]: 31    28
19    27
24    22
27    14
17    13
26    12
23    12
21     8
30     8
25     8
38     7
28     7
37     6
16     6
22     4
15     3
18     3
29     3
20     3
14     2
49     1
47     1
32     1
33     1
34     1
35     1
36     1
45     1
13     1
Name: 23, dtype: int64
```

Is there any attribute noise, erroneous (at data entry, violation of known data constraints) ?

Is there any inconsistent data ?

```
In [35]: df[2].value_counts()
Out[35]: toyota           32
nissan           18
mazda            17
honda            13
mitsubishi       13
subaru           12
volkswagen       12
volvo            11
peugot           11
dodge             9
mercedes-benz     8
bmw               8
plymouth          7
audi              7
saab              6
porsche           5
isuzu             4
jaguar            3
chevrolet         3
alfa-romero       3
renault           2
mercury           1
Name: 2, dtype: int64
```

Attribute: Identify Missing Values

```
In [50]: df[5].value_counts()

Out[50]: four    114
         two      89
         ?         2
         Name: 5, dtype: int64
```

Attribute 5 num-of-doors: four, two
Attribute 1 normalized-losses:
continuous from 65 to 256.

```
In [66]: df[1].value_counts()

Out[66]: ?      41
         161    11
         91      8
         150     7
         128     6
         104     6
         134     6
         168     5
         95      5
         103     5
         85      5
         65      5
         94      5
         102     5
         74      5
         93      4
         118     4
         106     4
         148     4
         122     4
         125     3
         83      3
         101     3
         115     3
         137     3
         154     3
         129     2
         164     2
         145     2
         87      2
```

Attribute: Identify Outliers

```
for i in range(len(df.columns)):
    if (df[i].dtypes in ['int64','float64']):
        print('\nAttribute-',i,':',df[i].dtypes)
        Q1=df[i].quantile(0.25)
        print('Q1',Q1)
        Q3=df[i].quantile(0.75)
        print('Q3',Q3)
        IQR=Q3-Q1
        print('IQR',IQR)
        min=df[i].min()
        max=df[i].max()
        min_IQR=Q1-1.5*IQR
        max_IQR=Q3+1.5*IQR
        if (min<min_IQR):
            print('Low outlier is found')
        if (max>max_IQR):
            print('High outlier is found')
```

Attribute: Identify Outliers

```
Attribute- 0 : int64
Q1 0.0
Q3 2.0
IQR 2.0

Attribute- 9 : float64
Q1 94.5
Q3 102.4
IQR 7.900000000000006
High outlier is found >  114.25
```

```
Attribute- 10 : float64
Q1 166.3
Q3 183.1
IQR 16.799999999999983
Low outlier is found <  141.10

Attribute- 11 : float64
Q1 64.1
Q3 66.9
IQR 2.8000000000000114
High outlier is found >  71.10
```

```
Attribute- 12 : float64
Q1 52.0
Q3 55.5
IQR 3.5

Attribute- 13 : int64
Q1 2145.0
Q3 2935.0
IQR 790.0
```

```
Attribute- 16 : int64
Q1 97.0
Q3 141.0
IQR 44.0
High outlier is found >  207.0

Attribute- 20 : float64
Q1 8.6
Q3 9.4
IQR 0.8000000000000007
Low outlier is found <  7.3999
High outlier is found >  10.60
```

```
Attribute- 23 : int64
Q1 19.0
Q3 30.0
IQR 11.0
High outlier is found >  46.5

Attribute- 24 : int64
Q1 25.0
Q3 34.0
IQR 9.0
High outlier is found >  47.5
```

```
In [30]: df.describe()
```

Out[30]:

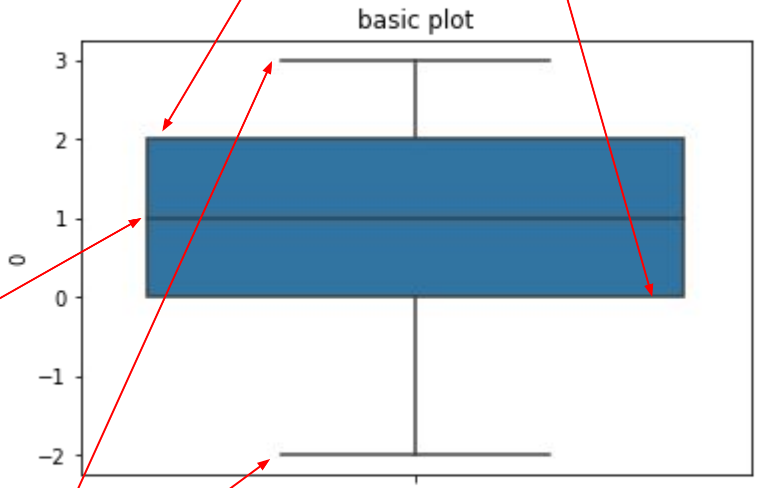
	0	9	10	11	12	13	16	20	23	24
count	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000	205.000000
mean	0.834146	98.756585	174.049268	65.907805	53.724878	2555.565854	126.907317	10.142537	25.219512	30.751220
std	1.245307	6.021776	12.337289	2.145204	2.443522	520.680204	41.642693	3.972040	6.542142	6.886443
min	-2.000000	86.600000	141.100000	60.300000	47.800000	1488.000000	61.000000	7.000000	13.000000	16.000000
25%	0.000000	94.500000	166.300000	64.100000	52.000000	2145.000000	97.000000	8.600000	19.000000	25.000000
50%	1.000000	97.000000	173.200000	65.500000	54.100000	2414.000000	120.000000	9.000000	24.000000	30.000000
75%	2.000000	102.400000	183.100000	66.900000	55.500000	2935.000000	141.000000	9.400000	30.000000	34.000000
max	3.000000	120.900000	208.100000	72.300000	59.800000	4066.000000	326.000000	23.000000	49.000000	54.000000

BoxPlot

The end of the box shows the upper and lower quartiles.

A boxplot is a standardized way of displaying the distribution of data based on a five number summary (“minimum”, first quartile (Q1), median, third quartile (Q3), and “maximum”).

- The line that divides the box into 2 parts represents the median of the data.



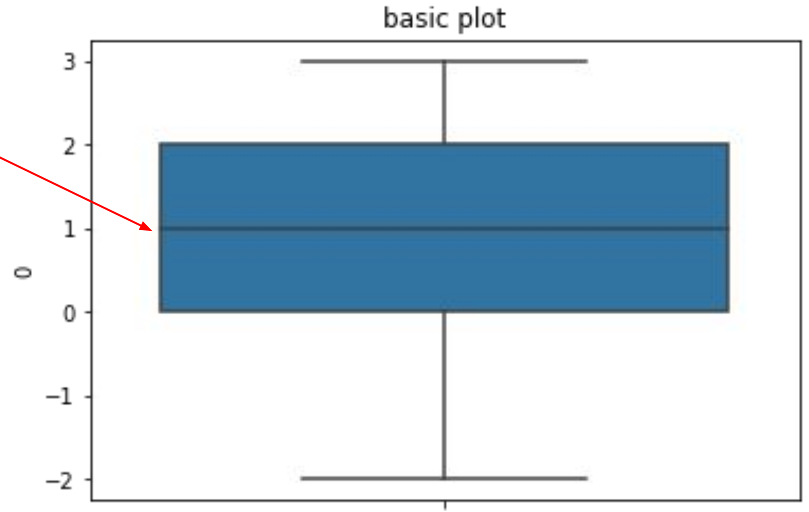
The extreme lines shows the highest and lowest value excluding outliers.

BoxPlot: Example

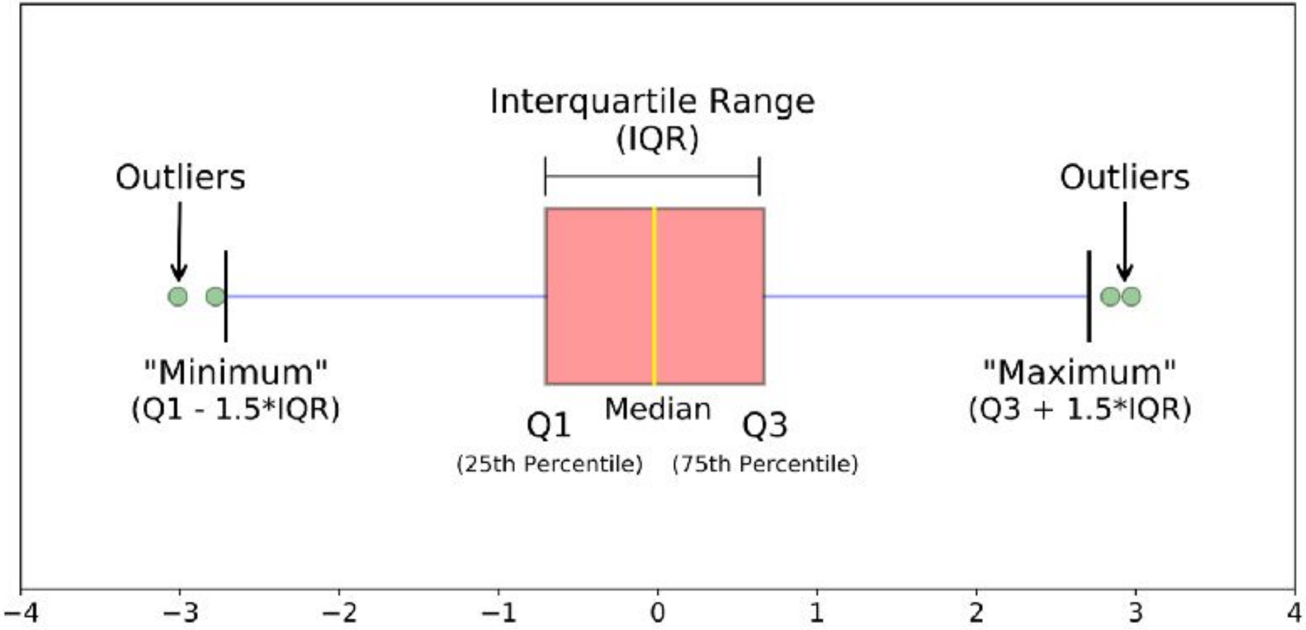
```
In [67]: df[0].describe()

Out[67]: count    205.000000
         mean       0.834146
         std        1.245307
         min       -2.000000
         25%        0.000000
         50%        1.000000
         75%        2.000000
         max        3.000000
         Name: 0, dtype: float64
```

```
In [68]: df[0].value_counts()

Out[68]:  0    67
          1    54
          2    32
          3    27
         -1    22
         -2     3
         Name: 0, dtype: int64
```

```
import seaborn as sns

# Make boxplot for one group only
sns.boxplot(y=df[0]).set_title('basic plot')
```



BoxPlot with Outliers

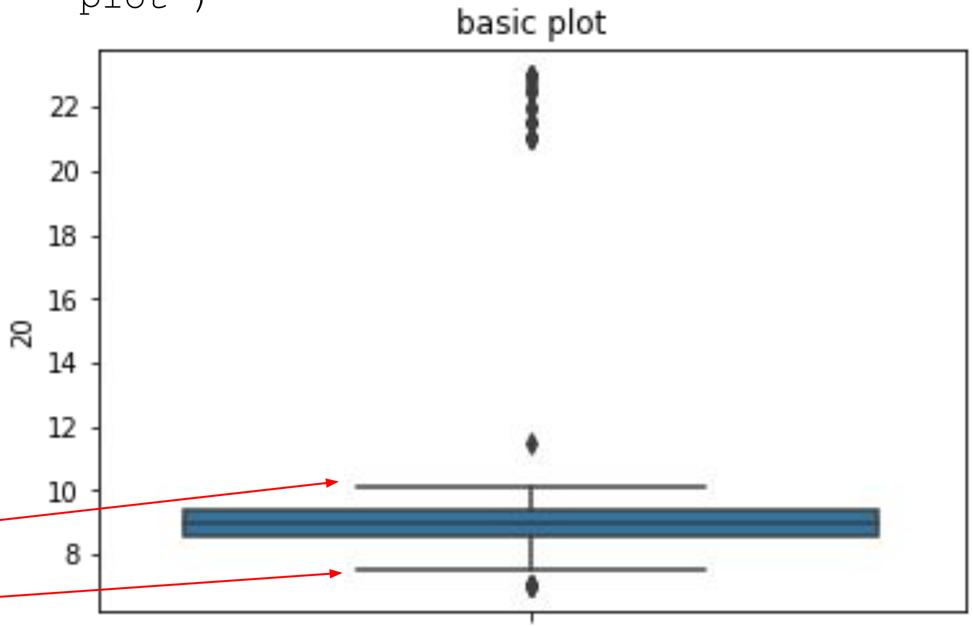


https://towardsdatascience.com/understanding-boxplots-5e2df7bcbd51

BoxPlot with Outliers: Example

```
In [80]: df[20].describe()

Out[80]: count    205.000000
         mean      10.142537
         std        3.972040
         min        7.000000
         25%        8.600000
         50%        9.000000
         75%        9.400000
         max       23.000000
         Name: 20, dtype: float64
```

```
import seaborn as sns
sns.boxplot(y=df[20]).set_title('basic plot')
```



IQR = 9.4-8.6=0.8

Max = Q3+1.5*IQR=9.4+1.2=10.6

Min = Q1-1.5*IQR=8.6-1.2=7.4

Correlation Coefficient

- ***Correlation coefficients*** are a quantitative measure that describe the strength of association/relationship between two variables.
- The correlation between two sets of data tells us about how they move together. Would changing one help us predict the other?

$$\rho_{XY} = \frac{\sigma_{XY}}{\sigma_X \sigma_Y}$$

$$\sigma_X^2 = E(X^2) - {\mu_X}^2$$
$$\sigma_Y^2 = E(Y^2) - {\mu_Y}^2$$

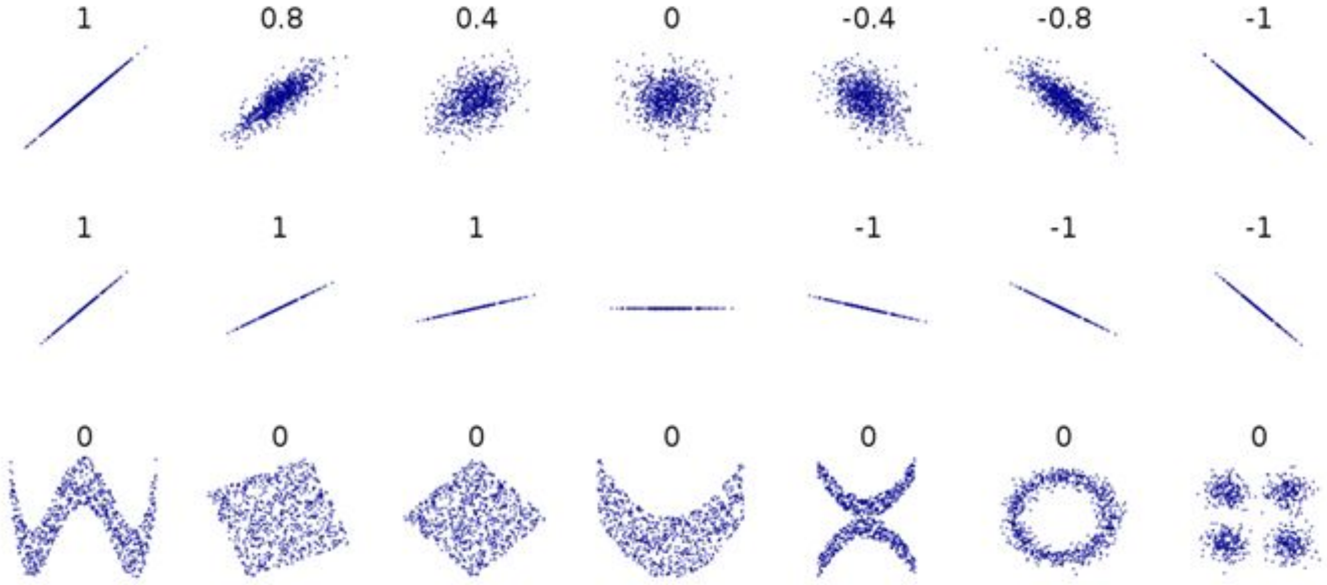
$$\sigma_{XY} = E(XY) - \mu_X \mu_Y$$

$$\sigma_{XY} = E[(X - \mu_X)(Y - \mu_Y)]$$

Correlation Coefficient: Notes

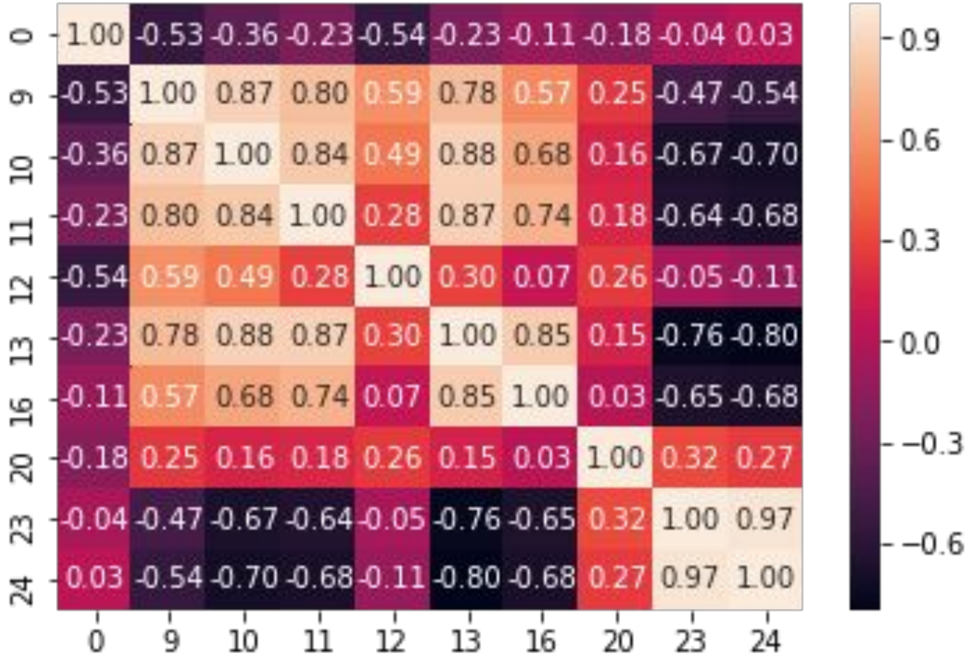
- Correlation coefficient will lie between -1 and 1
- The greater the absolute value (closer to -1 or 1), the stronger the relationship between the variables:
  - The strongest correlation is a -1 or a 1
  - The weakest correlation is a 0
- A positive correlation means that as one variable increases, the other one tends to increase as well
- A negative correlation means that as one variable increases, the other one tends to decrease

Correlation Coefficient



Correlation Coefficient Heatmap

```
import seaborn as sns

corr = df.corr()

sns.heatmap(corr,
annot=True, fmt='.2f')
```



Categorical Encoding

01 Direct encoding

- For attributes that represent numeric values
- Example: num_doors: two, four → {2,4}

02 Label encoding

- For ordinal attributes
- Example: size {S,M,L} → {1,2,3}.

03 One-hot encoding

- For multivalues attributes
- Example: city {BDO, JKT, ...} will be represented by city_BDO, city_JKT, etc

Categorical Encoding: Examples

atr1	atr2	atr3
two	S	BDO
four	M	JKT
one	L	DPS
four	M	BDO

Atr1: direct encoding
Atr2: label encoding
Atr3: one-hot encoding

atr1	atr2	atr3_ BDO	atr3_ JKT	atr3_ DPS
2	1	1	0	0
4	2	0	1	0
1	3	0	0	1
4	2	1	0	0

atr1	atr2	atr3
two	S	BDO
four	M	JKT
one	L	DPS
four	M	BDO

Atr1: direct encoding
Atr2, Atr3: one-hot encoding

atr1	atr2_S	atr2_M	atr2_L	atr3_BDO	atr3_JKT	atr3_DPS
2	1	0	0	1	0	0
4	0	1	0	0	1	0
1	0	0	1	0	0	1
4	0	1	0	1	0	0

Encoding Langsung

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500

```
df[5]=df[5].replace({'two': 2, 'four': 4})
```

	0	1	2	3	4	5
0	3	?	alfa-romero	gas	std	2
1	3	?	alfa-romero	gas	std	2
2	1	?	alfa-romero	gas	std	2

Label Encoding in Python

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500

```
for i in range(len(df.columns)):
    if (df[i].dtypes=='object'):
        df[i] = df[i].astype('category')
        df[i] = df[i].cat.codes
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	122.0	0	1	0	1	0	2	0	88.6	...	130	5	3.47	2.68	9.0	111.0	5000.0	21	27	13495.0
1	3	122.0	0	1	0	1	0	2	0	88.6	...	130	5	3.47	2.68	9.0	111.0	5000.0	21	27	16500.0
2	1	122.0	0	1	0	1	2	2	0	94.5	...	152	5	2.68	3.47	9.0	154.0	5000.0	19	26	16500.0

Label Encoding: Label Encoder

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
for i in range(len(df.columns)):
    if (df[i].dtypes=='object'):
        le.fit(df[i])
        df[i] = le.transform(df[i])
df.head()
```

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	NaN	0	1	0	2	0	2	0	88.6	...	130	5	3.47	2.68	9.0	111.0	5000.0	21	27	13495.0
1	3	NaN	0	1	0	2	0	2	0	88.6	...	130	5	3.47	2.68	9.0	111.0	5000.0	21	27	16500.0
2	1	NaN	0	1	0	2	2	2	0	94.5	...	152	5	2.68	3.47	9.0	154.0	5000.0	19	26	16500.0

One-hot Encoding

pandas.**get_dummies**(*data*, *prefix=None*, *prefix_sep='_'*, *dummy_na=False*, *columns=None*, *sparse=False*, *drop_first=False*, *dtype=None*)

Convert categorical variable into dummy/indicator variables

```
pd.get_dummies(df, columns=['col1', 'col2'])
```

One-hot Encoding

	0	1	2	3	4	5	6	7	8	9	...	16	17	18	19	20	21	22	23	24	25
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	13495
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi	3.47	2.68	9.0	111	5000	21	27	16500
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi	2.68	3.47	9.0	154	5000	19	26	16500

```
df_encode = pd.get_dummies(data=df, columns=[2])
df_encode.head()
```

	0	1	3	4	5	6	7	8	9	10	...	2_nissan	2_peugot	2_plymouth	2_porsche	2_renault	2_saab	2_subaru	2_
0	3	NaN	gas	std	two	convertible	rwd	front	88.6	168.8	...	0	0	0	0	0	0	0	
1	3	NaN	gas	std	two	convertible	rwd	front	88.6	168.8	...	0	0	0	0	0	0	0	
2	1	NaN	gas	std	two	hatchback	rwd	front	94.5	171.2	...	0	0	0	0	0	0	0	