AI VIET NAM
@aivietnam.edu.vn

Data Visualization and Analysis Using Pandas

Quang-Vinh Dinh
Ph.D. in Computer Science

Year 2024

Objectives

Dataframe in Pandas

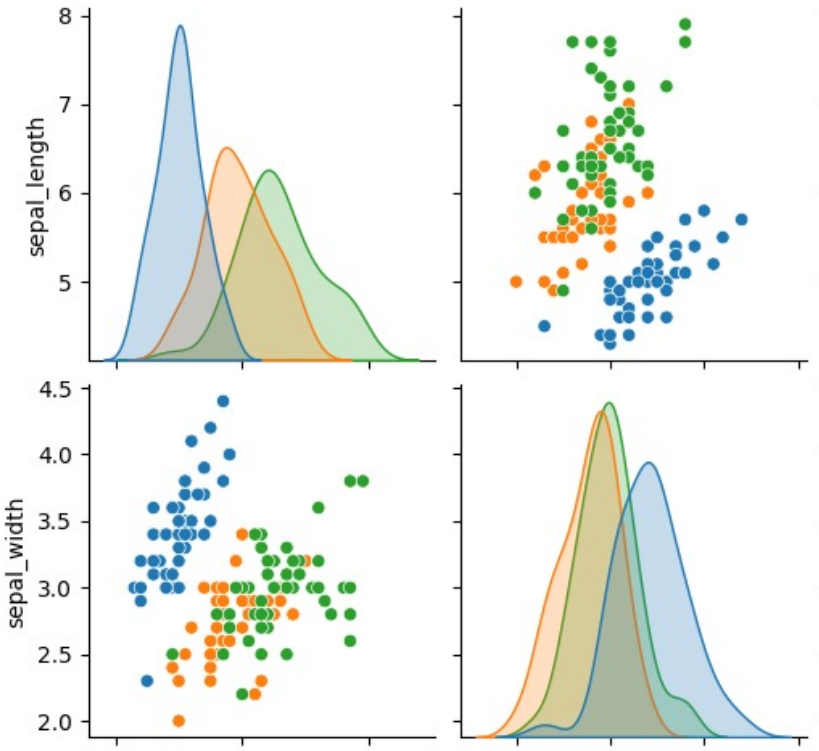
df.columns

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

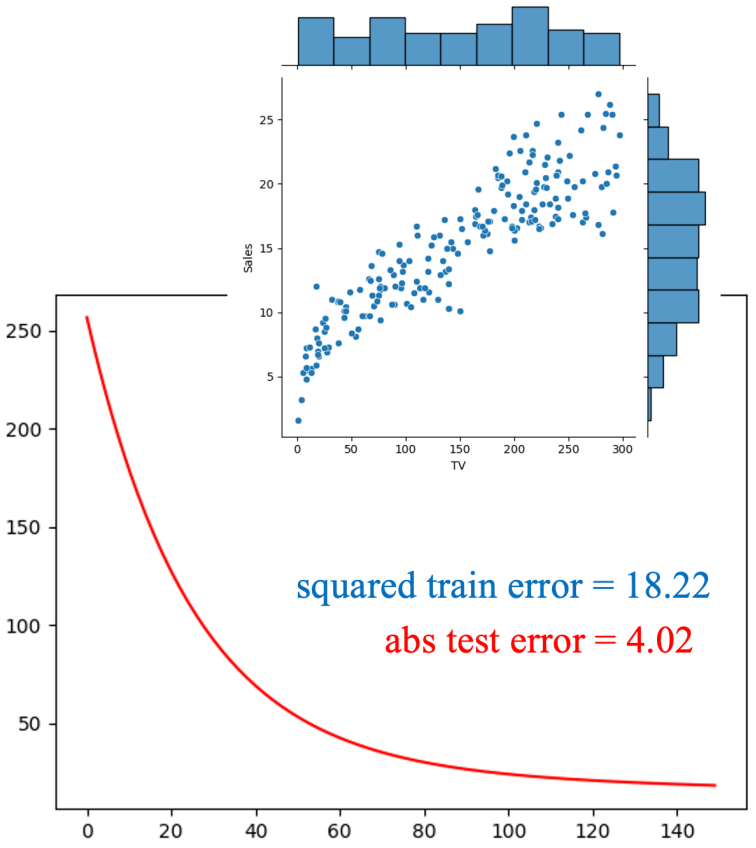
df.index

df['Newspaper']
df.Newspaper

Visualization



Case Studies



Outline

SECTION 1

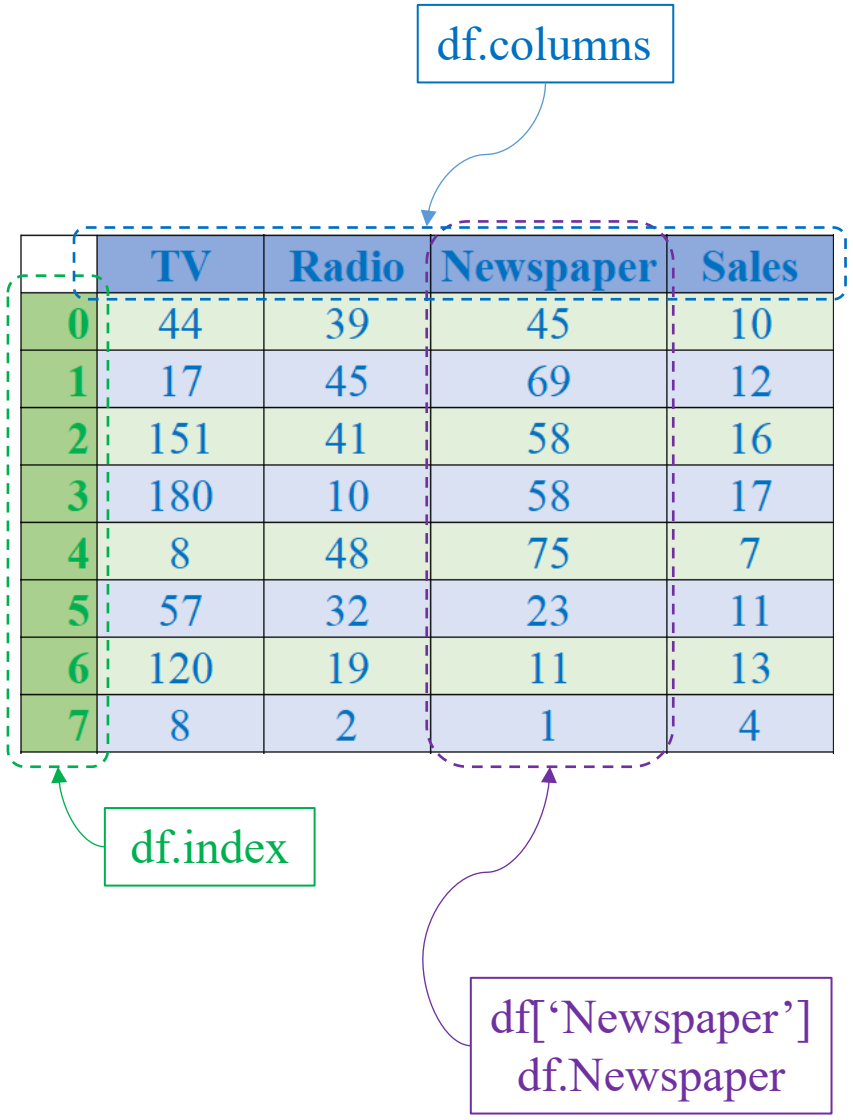
DataFrame in Pandas

SECTION 2

Visualization

SECTION 3

Case Studies



Data Frame

❖ Create a dataframe

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
4  print(df)
5
6  df_t = df.T
7  print(df_t)
```

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.columns

df.index

df['Newspaper']
df.Newspaper

	0	1	2	3	4	5	6	7
TV	44	17	151	180	8	57	120	8
Radio	39	45	41	10	48	32	19	2
Newspaper	45	69	58	58	75	23	11	1
Sales	10	12	16	17	7	11	13	4

df_t.columns

df_t.index

Data Frame

❖ Sorting

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
4  df.sort_values('Sales')
```

	TV	Radio	Newspaper	Sales
7	8	2	1	4
4	8	48	75	7
0	44	39	45	10
5	57	32	23	11
1	17	45	69	12
6	120	19	11	13
2	151	41	58	16
3	180	10	58	17

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
4  df.sort_values(['Newspaper', 'Sales'])
```

	TV	Radio	Newspaper	Sales
7	8	2	1	4
6	120	19	11	13
5	57	32	23	11
0	44	39	45	10
2	151	41	58	16
3	180	10	58	17
1	17	45	69	12
4	8	48	75	7

Data Frame

❖ Add a column

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
4  df['ID'] = list(range(2, 10))
5  df = df.set_index('ID')
```

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.columns

df.index

df['Newspaper']

	TV	Radio	Newspaper	Sales	ID
0	44	39	45	10	2
1	17	45	69	12	3
2	151	41	58	16	4
3	180	10	58	17	5
4	8	48	75	7	6
5	57	32	23	11	7
6	120	19	11	13	8
7	8	2	1	4	9

ID	TV	Radio	Newspaper	Sales
2	44	39	45	10
3	17	45	69	12
4	151	41	58	16
5	180	10	58	17
6	8	48	75	7
7	57	32	23	11
8	120	19	11	13
9	8	2	1	4

Example 1

❖ Weather dataset

Formatted Date	Temperature (C)
2006-01-01 00:00:00.000 +0100	0.577777778
2006-01-01 01:00:00.000 +0100	1.161111111
2006-01-01 02:00:00.000 +0100	1.666666667
2006-01-01 03:00:00.000 +0100	1.711111111
2006-01-01 04:00:00.000 +0100	1.183333333
2006-01-01 05:00:00.000 +0100	1.205555556
2006-01-01 06:00:00.000 +0100	2.222222222
2006-01-01 07:00:00.000 +0100	2.072222222
2006-01-01 08:00:00.000 +0100	2.2
2006-01-01 09:00:00.000 +0100	2.733333333
2006-01-01 10:00:00.000 +0100	2.788888889
2006-01-01 11:00:00.000 +0100	3.822222222
2006-01-01 12:00:00.000 +0100	4.911111111
2006-01-01 13:00:00.000 +0100	6.205555556
2006-01-01 14:00:00.000 +0100	7.438888889
2006-01-01 15:00:00.000 +0100	6.95

temperature = 0.57

datetime = ‘2006-12-01 00’

➡ Model

Data to teach a model

temperature = 0.57

year = 2006

month = 12

day = 1

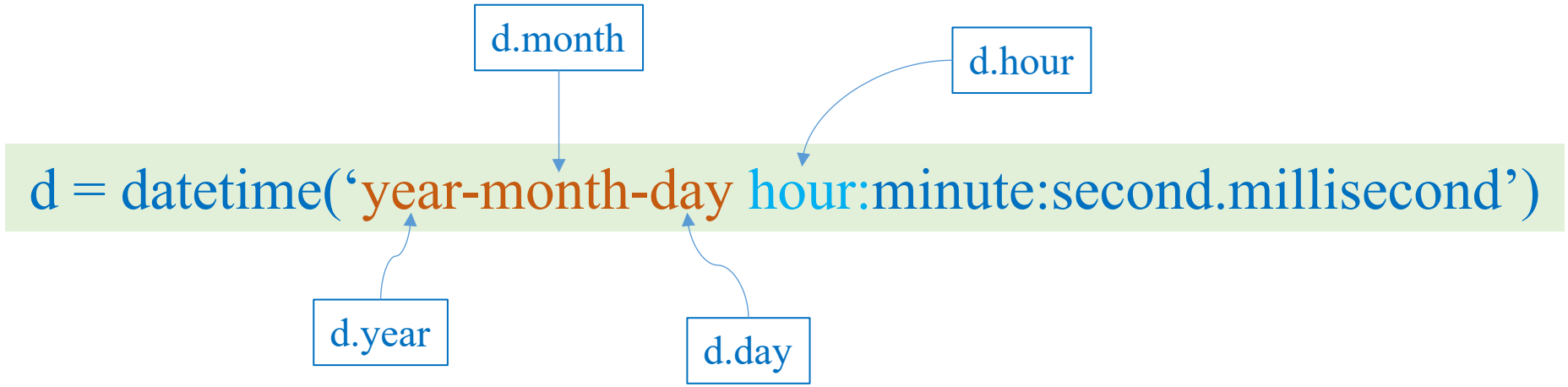
hour = 0

➡ Model

Which one is easier for the model to learn?

Example 1

❖ Datetime




```
datetime = pd.to_datetime('2006-12-01 00:00:00.000')
print(datetime)
print(datetime.year)
print(datetime.month)
print(datetime.day)
print(datetime.hour)
```

```
2006-12-01 00:00:00
2006
12
1
0
```

```
df = pd.DataFrame({
    'year': [datetime.year],
    'month': [datetime.month],
    'day': [datetime.day],
    'hour': [datetime.hour]
})

# print
print(df)
```

✓ 0.0s

```
   year  month  day  hour
0  2006     12    1     0
```

AI VIET NAM
@aivietnam.edu.vn

Example 1

```
df['Formatted Date'] = pd.to_datetime(df['Formatted Date'],
                                      utc=True)
df['month'] = df['Formatted Date'].dt.month
df['year'] = df['Formatted Date'].dt.year
df['day'] = df['Formatted Date'].dt.day
df['hour'] = df['Formatted Date'].dt.hour
```

❖ Weather dataset

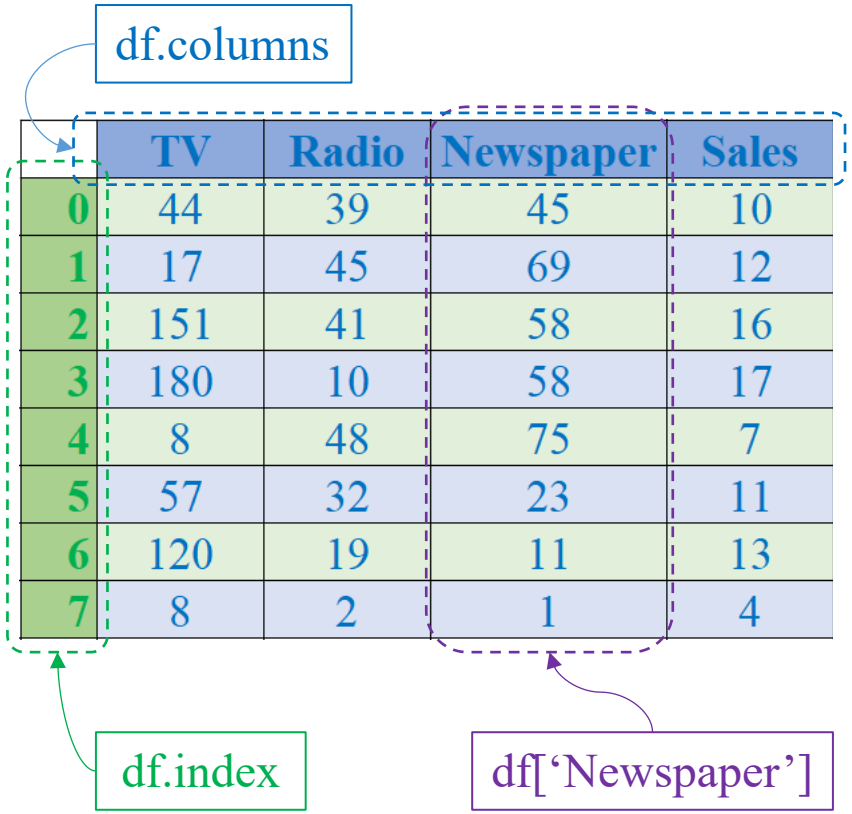
Formatted Date	Temperature (C)
2006-01-01 00:00:00.000 +0100	0.577777778
2006-01-01 01:00:00.000 +0100	1.161111111
2006-01-01 02:00:00.000 +0100	1.666666667
2006-01-01 03:00:00.000 +0100	1.711111111
2006-01-01 04:00:00.000 +0100	1.183333333
2006-01-01 05:00:00.000 +0100	1.205555556
2006-01-01 06:00:00.000 +0100	2.222222222
2006-01-01 07:00:00.000 +0100	2.072222222
2006-01-01 08:00:00.000 +0100	2.2
2006-01-01 09:00:00.000 +0100	2.733333333
2006-01-01 10:00:00.000 +0100	2.788888889
2006-01-01 11:00:00.000 +0100	3.822222222
2006-01-01 12:00:00.000 +0100	4.911111111
2006-01-01 13:00:00.000 +0100	6.205555556
2006-01-01 14:00:00.000 +0100	7.438888889
2006-01-01 15:00:00.000 +0100	6.95

Formatted Date	Temperature (C)	month	year	day	hour
2005-12-31 23:00:00+00:00	0.577777778	12	2005	31	23
2006-01-01 00:00:00+00:00	1.161111111	1	2006	1	0
2006-01-01 01:00:00+00:00	1.666666667	1	2006	1	1
2006-01-01 02:00:00+00:00	1.711111111	1	2006	1	2
2006-01-01 03:00:00+00:00	1.183333333	1	2006	1	3
2006-01-01 04:00:00+00:00	1.205555556	1	2006	1	4
2006-01-01 05:00:00+00:00	2.222222222	1	2006	1	5
2006-01-01 06:00:00+00:00	2.072222222	1	2006	1	6
2006-01-01 07:00:00+00:00	2.2	1	2006	1	7
2006-01-01 08:00:00+00:00	2.733333333	1	2006	1	8
2006-01-01 09:00:00+00:00	2.788888889	1	2006	1	9
2006-01-01 10:00:00+00:00	3.822222222	1	2006	1	10
2006-01-01 11:00:00+00:00	4.911111111	1	2006	1	11
2006-01-01 12:00:00+00:00	6.205555556	1	2006	1	12
2006-01-01 13:00:00+00:00	7.438888889	1	2006	1	13
2006-01-01 14:00:00+00:00	6.95	1	2006	1	14

Data Frame

❖ Delete a row

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
```



	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.drop(1, axis=0)

	TV	Radio	Newspaper	Sales
0	44	39	45	10
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.drop([1, 2, 3], axis=0)

	TV	Radio	Newspaper	Sales
0	44	39	45	10
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

Data Frame

❖ Delete a row

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
```

df.columns

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.index

df['Newspaper']

	TV	Newspaper	Sales
0	44	45	10
1	17	69	12
2	151	58	16
3	180	58	17
4	8	75	7
5	57	23	11
6	120	11	13
7	8	1	4

df.drop('Radio', axis=1)

	TV	Newspaper
0	44	45
1	17	69
2	151	58
3	180	58
4	8	75
5	57	23
6	120	11
7	8	1

df.drop(['Radio', 'Sales'], axis=1)

Example 2

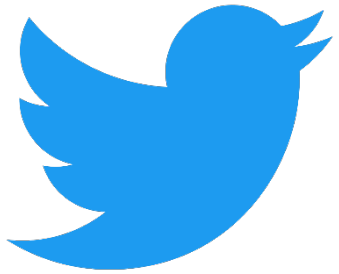
❖ Tweets dataset

- Training samples: 7613
- Total Number of disaster tweets: 4342
- Total Number of non-disaster tweets: 3271

id	keyword	location	text	target
48	ablaze	Birmingham	@bbcmtd Wholesale Markets ablaze http://t.co/lH	1
49	ablaze	Est. September 2012 - Bri	We always try to bring the heavy. #metal #RT htt	0
144	accident	UK	.@NorwayMFA #Bahrain police had previously c	1
145	accident	Nairobi, Kenya	I still have not heard Church Leaders of Kenya co	0
146	aftershock	Instagram - @heyimginog	@afterShock_DeLo scuf ps live and the game... c	0
1669	bombing	London	Japan marks 70th anniversary of Hiroshima atomi	1
1670	bombing	United States	Japan marks 70th anniversary of Hiroshima atomi	1
6536	injury	beijing .China	Rory McIlroy to Test Ankle Injury in Weekend P	0
6537	injury	MISSING	CLEARED:incident with injury:I-495 inner loop	1

Example 2

❖ Tweets dataset



id	keyword	location	text	target
48	ablaze	Birmingham	@bbcmtd Wholesale Markets ablaze http://t.co/lH	1
49	ablaze	Est. September 2012 - Bri	We always try to bring the heavy. #metal #RT htt	0
144	accident	UK	.@NorwayMFA #Bahrain police had previously	1
145	accident	Nairobi, Kenya	I still have not heard Church Leaders of Kenya co	0
146	aftershock	Instagram - @heyimginog	@afterShock_DeLo scuf ps live and the game... c	0
1669	bombing	London	Japan marks 70th anniversary of Hiroshima atomi	1
1670	bombing	United States	Japan marks 70th anniversary of Hiroshima atomi	1
6536	injury	beijing .China	Rory McIlroy to Test Ankle Injury in Weekend P	0
6537	injury	MISSING	CLEARED:incident with injury:I-495 inner loop	1

```python
import pandas as pd

df = pd.read_csv("path_to_cvs_file")
df.drop(["id", "keyword", "location"],
        axis=1,
        inplace=True)
```

text	target
@bbcmtd Wholesale Markets ablaze http://t.co/lH	1
We always try to bring the heavy. #metal #RT htt	0
.@NorwayMFA #Bahrain police had previously	1
I still have not heard Church Leaders of Kenya co	0
@afterShock_DeLo scuf ps live and the game... c	0
Japan marks 70th anniversary of Hiroshima atomi	1
Japan marks 70th anniversary of Hiroshima atomi	1
Rory McIlroy to Test Ankle Injury in Weekend P	0
CLEARED:incident with injury:I-495 inner loop	1

Data Frame

❖ Get values

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
```

df.columns

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.index

df['Newspaper']

df.Radio
df['Radio']

	Radio
0	39
1	45
2	41
3	10
4	48
5	32
6	19
7	2

df[['Radio', 'Sales']]

	Radio	Sales
0	39	10
1	45	12
2	41	16
3	10	17
4	48	7
5	32	11
6	19	13
7	2	4

df[1:2]

	TV	Radio	Newspaper	Sales
1	17	45	69	12

df[4:]

	TV	Radio	Newspaper	Sales
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df[1:4:2]

	TV	Radio	Newspaper	Sales
1	17	45	69	12
3	180	10	58	17

Data Frame

❖ Groupby

```
1  import pandas as pd
2
3  df = pd.read_csv('weatherHistory_simple.csv')
4  df.describe()
5
6  df.groupby('Precip Type').mean('Temperature (C)')
7  df['Fahrenheit'] = df['Temperature (C)']*9/5.0 + 32
```

	Temperature (C)
count	10
mean	6.25
std	7.63
min	-2.1
0.25	-0.92
0.5	6.35
0.75	13.52
max	14.7

Precip Type	Temperature (C)
rain	8.972222
snow	-2.038889

	Precip Type	Temperature (C)
0	rain	12.2
1	rain	13.8
2	rain	14.7
3	rain	13.8
4	rain	12.7
5	rain	0.5
6	snow	-0.4
7	snow	-1.1
8	snow	-1.6
9	snow	-2.1

	Precip Type	Temperature (C)	Fahrenheit
0	rain	12.2	53.96
1	rain	13.8	56.84
2	rain	14.7	58.46
3	rain	13.8	56.84
4	rain	12.7	54.86
5	rain	0.5	32.9
6	snow	-0.4	31.28
7	snow	-1.1	30.02
8	snow	-1.6	29.12
9	snow	-2.1	28.22

Example 3: Credit Card Fraud Detection

The dataset contains transactions made by credit cards in September 2013 by European cardholders.

Feature 'Class' is the response variable, and it takes value 1 in case of fraud and 0 otherwise

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Recall} = \frac{TP}{TP + FN} \qquad \text{Precision} = \frac{TP}{TP + FP}$$

```
1  import pandas as pd
2
3  df = pd.read_csv('creditcard.csv')
4  df.groupby('Class').size()
```

Class	#samples
0	284315
1	492

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0	-1.3598	-0.0728	2.5363	1.3782	-0.3383	0.4624	0.2396	0.0987	0.3638	...	-0.0183	0.2778	-0.1105	0.0669	0.1285	-0.1891	0.1336	-0.0211	149.62	0
1	0	1.1919	0.2662	0.1665	0.4482	0.06	-0.0824	-0.0788	0.0851	-0.2554	...	-0.2258	-0.6387	0.1013	-0.3398	0.1672	0.1259	-0.009	0.0147	2.69	0
2	1	-1.3584	-1.3402	1.7732	0.3798	-0.5032	1.8005	0.7915	0.2477	-1.5147	...	0.248	0.7717	0.9094	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
3	1	-0.9663	-0.1852	1.793	-0.8633	-0.0103	1.2472	0.2376	0.3774	-1.387	...	-0.1083	0.0053	-0.1903	-1.1756	0.6474	-0.2219	0.0627	0.0615	123.5	0
4	2	-1.1582	0.8777	1.5487	0.403	-0.4072	0.0959	0.5929	-0.2705	0.8177	...	-0.0094	0.7983	-0.1375	0.1413	-0.206	0.5023	0.2194	0.2152	69.99	0
5	2	-0.426	0.9605	1.1411	-0.1683	0.421	-0.0297	0.4762	0.2603	-0.5687	...	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.0811	3.67	0
6	4	1.2297	0.141	0.0454	1.2026	0.1919	0.2727	-0.0052	0.0812	0.465	...	-0.1677	-0.2707	-0.1541	-0.7801	0.7501	-0.2572	0.0345	0.0052	4.99	0
7	7	-0.6443	1.418	1.0744	-0.4922	0.9489	0.4281	1.1206	-3.8079	0.6154	...	1.9435	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
8	7	-0.8943	0.2862	-0.1132	-0.2715	2.6696	3.7218	0.3701	0.8511	-0.392	...	-0.0734	-0.2681	-0.2042	1.0116	0.3732	-0.3842	0.0117	0.1424	93.2	0
9	9	-0.3383	1.1196	1.0444	-0.2222	0.4994	-0.2468	0.6516	0.0695	-0.7367	...	-0.2469	-0.6338	-0.1208	-0.3851	-0.0697	0.0942	0.2462	0.0831	3.68	0

Example 3

❖ **Ignore the problem**

Experimental results

$\mathrm{TP} = 57$

$FN = 18$

$Recall = 0.76$

```python
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(256, activation="relu",
                       input_shape=(30,)),
                       keras.layers.Dense(256, activation="relu"),
                       keras.layers.Dropout(0.3),
                       keras.layers.Dense(1, activation="sigmoid")])

model.compile(optimizer=keras.optimizers.Adam(1e-2),
              loss="binary_crossentropy", metrics=metrics)

model.fit(train_features, train_targets,
          batch_size=2048, epochs=20,
          validation_data=(val_features, val_targets))
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0	-1.3598	-0.0728	2.5363	1.3782	-0.3383	0.4624	0.2396	0.0987	0.3638	...	-0.0183	0.2778	-0.1105	0.0669	0.1285	-0.1891	0.1336	-0.0211	149.62	0
1	0	1.1919	0.2662	0.1665	0.4482	0.06	-0.0824	-0.0788	0.0851	-0.2554	...	-0.2258	-0.6387	0.1013	-0.3398	0.1672	0.1259	-0.009	0.0147	2.69	0
2	1	-1.3584	-1.3402	1.7732	0.3798	-0.5032	1.8005	0.7915	0.2477	-1.5147	...	0.248	0.7717	0.9094	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
3	1	-0.9663	-0.1852	1.793	-0.8633	-0.0103	1.2472	0.2376	0.3774	-1.387	...	-0.1083	0.0053	-0.1903	-1.1756	0.6474	-0.2219	0.0627	0.0615	123.5	0
4	2	-1.1582	0.8777	1.5487	0.403	-0.4072	0.0959	0.5929	-0.2705	0.8177	...	-0.0094	0.7983	-0.1375	0.1413	-0.206	0.5023	0.2194	0.2152	69.99	0
5	2	-0.426	0.9605	1.1411	-0.1683	0.421	-0.0297	0.4762	0.2603	-0.5687	...	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.0811	3.67	0
6	4	1.2297	0.141	0.0454	1.2026	0.1919	0.2727	-0.0052	0.0812	0.465	...	-0.1677	-0.2707	-0.1541	-0.7801	0.7501	-0.2572	0.0345	0.0052	4.99	0
7	7	-0.6443	1.418	1.0744	-0.4922	0.9489	0.4281	1.1206	-3.8079	0.6154	...	1.9435	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
8	7	-0.8943	0.2862	-0.1132	-0.2715	2.6696	3.7218	0.3701	0.8511	-0.392	...	-0.0734	-0.2681	-0.2042	1.0116	0.3732	-0.3842	0.0117	0.1424	93.2	0
9	9	-0.3383	1.1196	1.0444	-0.2222	0.4994	-0.2468	0.6516	0.0695	-0.7367	...	-0.2469	-0.6338	-0.1208	-0.3851	-0.0697	0.0942	0.2462	0.0831	3.68	0

Example 3

❖ **Class weights**

Experimental results

$TP = 68$

$FN = 7$

$Recall = 0.906$

```python
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(256, activation="relu",
                       input_shape=(30,)),
                       keras.layers.Dense(256, activation="relu"),
                       keras.layers.Dropout(0.3),
                       keras.layers.Dense(1, activation="sigmoid")])

model.compile(optimizer=keras.optimizers.Adam(1e-2),
              loss="binary_crossentropy", metrics=metrics)

weight_for_0 = total / (2.0*num_class_0)
weight_for_1 = total / (2.0*num_class_1)
class_weight = {0: weight_for_0, 1: weight_for_1}

model.fit(train_features, train_targets,
          batch_size=2048, epochs=20,
          validation_data=(val_features, val_targets),
          class_weight=class_weight)
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0	-1.3598	-0.0728	2.5363	1.3782	-0.3383	0.4624	0.2396	0.0987	0.3638	...	-0.0183	0.2778	-0.1105	0.0669	0.1285	-0.1891	0.1336	-0.0211	149.62	0
1	0	1.1919	0.2662	0.1665	0.4482	0.06	-0.0824	-0.0788	0.0851	-0.2554	...	-0.2258	-0.6387	0.1013	-0.3398	0.1672	0.1259	-0.009	0.0147	2.69	0
2	1	-1.3584	-1.3402	1.7732	0.3798	-0.5032	1.8005	0.7915	0.2477	-1.5147	...	0.248	0.7717	0.9094	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
3	1	-0.9663	-0.1852	1.793	-0.8633	-0.0103	1.2472	0.2376	0.3774	-1.387	...	-0.1083	0.0053	-0.1903	-1.1756	0.6474	-0.2219	0.0627	0.0615	123.5	0
4	2	-1.1582	0.8777	1.5487	0.403	-0.4072	0.0959	0.5929	-0.2705	0.8177	...	-0.0094	0.7983	-0.1375	0.1413	-0.206	0.5023	0.2194	0.2152	69.99	0
5	2	-0.426	0.9605	1.1411	-0.1683	0.421	-0.0297	0.4762	0.2603	-0.5687	...	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.0811	3.67	0
6	4	1.2297	0.141	0.0454	1.2026	0.1919	0.2727	-0.0052	0.0812	0.465	...	-0.1677	-0.2707	-0.1541	-0.7801	0.7501	-0.2572	0.0345	0.0052	4.99	0
7	7	-0.6443	1.418	1.0744	-0.4922	0.9489	0.4281	1.1206	-3.8079	0.6154	...	1.9435	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
8	7	-0.8943	0.2862	-0.1132	-0.2715	2.6696	3.7218	0.3701	0.8511	-0.392	...	-0.0734	-0.2681	-0.2042	1.0116	0.3732	-0.3842	0.0117	0.1424	93.2	0
9	9	-0.3383	1.1196	1.0444	-0.2222	0.4994	-0.2468	0.6516	0.0695	-0.7367	...	-0.2469	-0.6338	-0.1208	-0.3851	-0.0697	0.0942	0.2462	0.0831	3.68	0

Example 3

❖ **Focal loss**

Experimental results

$TP = 92$

$FN = 3$

$Recall = 0.96$

```python
from tensorflow import keras

model = keras.Sequential([
    keras.layers.Dense(256, activation="relu",
                       input_shape=(30,)),
                       keras.layers.Dense(256, activation="relu"),
                       keras.layers.Dropout(0.3),
                       keras.layers.Dense(1, activation="sigmoid")])

loss_func = tfa.losses.SigmoidFocalCrossEntropy(alpha=0.99,
                                                gamma=3.0)
model.compile(optimizer=keras.optimizers.Adam(1e-2),
              loss=loss_func, metrics=metrics)

model.fit(train_features, train_targets,
          batch_size=2048, epochs=20,
          validation_data=(val_features, val_targets))
```

	Time	V1	V2	V3	V4	V5	V6	V7	V8	V9	...	V21	V22	V23	V24	V25	V26	V27	V28	Amount	Class
0	0	-1.3598	-0.0728	2.5363	1.3782	-0.3383	0.4624	0.2396	0.0987	0.3638	...	-0.0183	0.2778	-0.1105	0.0669	0.1285	-0.1891	0.1336	-0.0211	149.62	0
1	0	1.1919	0.2662	0.1665	0.4482	0.06	-0.0824	-0.0788	0.0851	-0.2554	...	-0.2258	-0.6387	0.1013	-0.3398	0.1672	0.1259	-0.009	0.0147	2.69	0
2	1	-1.3584	-1.3402	1.7732	0.3798	-0.5032	1.8005	0.7915	0.2477	-1.5147	...	0.248	0.7717	0.9094	-0.6893	-0.3276	-0.1391	-0.0554	-0.0598	378.66	0
3	1	-0.9663	-0.1852	1.793	-0.8633	-0.0103	1.2472	0.2376	0.3774	-1.387	...	-0.1083	0.0053	-0.1903	-1.1756	0.6474	-0.2219	0.0627	0.0615	123.5	0
4	2	-1.1582	0.8777	1.5487	0.403	-0.4072	0.0959	0.5929	-0.2705	0.8177	...	-0.0094	0.7983	-0.1375	0.1413	-0.206	0.5023	0.2194	0.2152	69.99	0
5	2	-0.426	0.9605	1.1411	-0.1683	0.421	-0.0297	0.4762	0.2603	-0.5687	...	-0.2083	-0.5598	-0.0264	-0.3714	-0.2328	0.1059	0.2538	0.0811	3.67	0
6	4	1.2297	0.141	0.0454	1.2026	0.1919	0.2727	-0.0052	0.0812	0.465	...	-0.1677	-0.2707	-0.1541	-0.7801	0.7501	-0.2572	0.0345	0.0052	4.99	0
7	7	-0.6443	1.418	1.0744	-0.4922	0.9489	0.4281	1.1206	-3.8079	0.6154	...	1.9435	-1.0155	0.0575	-0.6497	-0.4153	-0.0516	-1.2069	-1.0853	40.8	0
8	7	-0.8943	0.2862	-0.1132	-0.2715	2.6696	3.7218	0.3701	0.8511	-0.392	...	-0.0734	-0.2681	-0.2042	1.0116	0.3732	-0.3842	0.0117	0.1424	93.2	0
9	9	-0.3383	1.1196	1.0444	-0.2222	0.4994	-0.2468	0.6516	0.0695	-0.7367	...	-0.2469	-0.6338	-0.1208	-0.3851	-0.0697	0.0942	0.2462	0.0831	3.68	0

Data Frame

❖ loc and iloc

loc for indexing by labels

iloc for indexing by positional index

```python
import pandas as pd

df = pd.read_csv('advertising_simple.csv')
df.set_index([['A', 'B', 'C', 'D',
               'F', 'G', 'H', 'I']],
             inplace=True)
```

df.columns

	TV	Radio	Newspaper	Sales
A	44	39	45	10
B	17	45	69	12
C	151	41	58	16
D	180	10	58	17
E	8	48	75	7
F	57	32	23	11
G	120	19	11	13
H	8	2	1	4

df.loc['C', 'Radio']

df.iloc[2, 1]

df.index

df.loc['F':'H', 'Radio':'Sales']

df.loc['B', :]

	TV	Radio	Newspaper	Sales
A	44	39	45	10
B	17	45	69	12
C	151	41	58	16
D	180	10	58	17
E	8	48	75	7
F	57	32	23	11
G	120	19	11	13
H	8	2	1	4

df.loc[['E', 'G'], :]

Pandas: Select values

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising_simple.csv')
```

df.columns

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df.index

```
# select * from df where df.Newspaper<30
df.loc[df.Newspaper<30]
```

	TV	Radio	Newspaper	Sales
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

```
# select * from df where Sales>10
df.loc[df.Sales>10]
```

	TV	Radio	Newspaper	Sales
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
5	57	32	23	11
6	120	19	11	13

	TV	Radio	Newspaper	Sales
5	57	32	23	11
6	120	19	11	13

```
# select * from df where Sales>10 and Newspaper<30
df.loc[(df.Sales>10) & (df.Newspaper<30)]
```

```
# sql: select Radio from df
df['Radio']
```

	Radio
0	39
1	45
2	41
3	10
4	48
5	32
6	19
7	2

Data Frame

❖ Concatenate data

```
1  import pandas as pd
2
3  df1 = pd.read_csv('advertising_p1.csv')
4  df2 = pd.read_csv('advertising_p2.csv')
```

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17

df1

	TV	Radio	Newspaper	Sales
0	8	48	75	7
1	57	32	23	11
2	120	19	11	13
3	8	2	1	4

df2

df3 = pd.concat([df1, df2])

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
0	8	48	75	7
1	57	32	23	11
2	120	19	11	13
3	8	2	1	4

df3

df4 = df3.reset_index(drop=True)

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df4

Data Frame

❖ Concatenate data

```
1  import pandas as pd
2
3  df1 = pd.read_csv('advertising_p1.csv')
4  df2 = pd.read_csv('advertising_p2.csv')
```

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17

df1

	TV	Radio	Newspaper	Sales
0	8	48	75	7
1	57	32	23	11
2	120	19	11	13
3	8	2	1	4

df2

df3 = pd.concat([df1, df2], ignore_index=True)

	TV	Radio	Newspaper	Sales
0	44	39	45	10
1	17	45	69	12
2	151	41	58	16
3	180	10	58	17
4	8	48	75	7
5	57	32	23	11
6	120	19	11	13
7	8	2	1	4

df3

Data Frame

Convert categories to numbers

```
1  import pandas as pd
2
3  df = pd.read_csv('weatherHistory_simple.csv')
4  df['Precip Type'] = pd.Categorical(df['Precip Type']).codes
```

	Precip Type	Temperature (C)
0	rain	12.2
1	rain	13.8
2	rain	14.7
3	rain	13.8
4	rain	12.7
5	rain	0.5
6	snow	-0.4
7	snow	-1.1
8	snow	-1.6
9	snow	-2.1

	Precip Type	Temperature (C)
0	0	12.2
1	0	13.8
2	0	14.7
3	0	13.8
4	0	12.7
5	0	0.5
6	1	-0.4
7	1	-1.1
8	1	-1.6
9	1	-2.1

Outline

SECTION 1

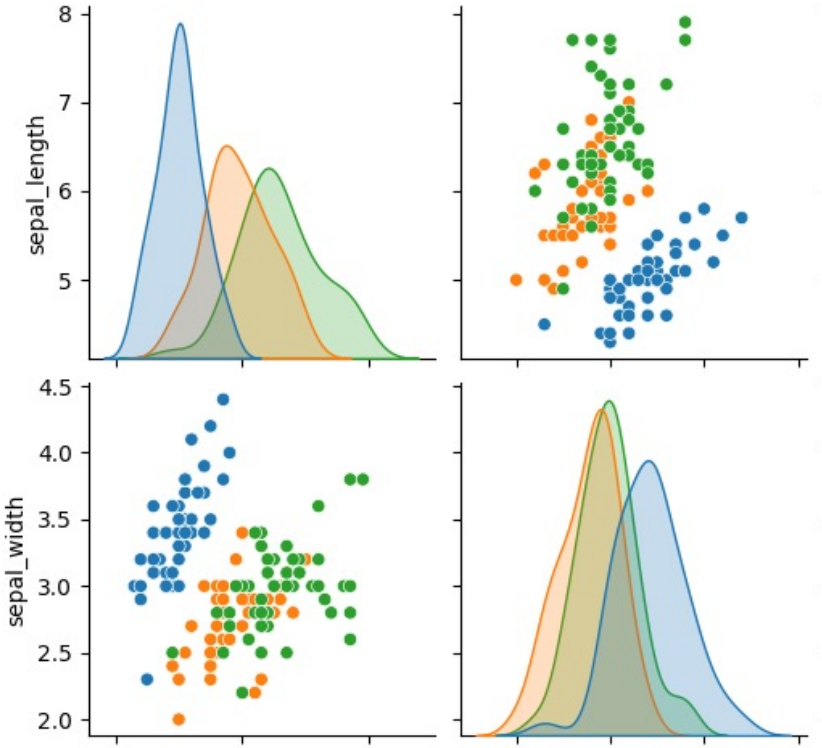
DataFrame in Pandas

SECTION 2

Visualization

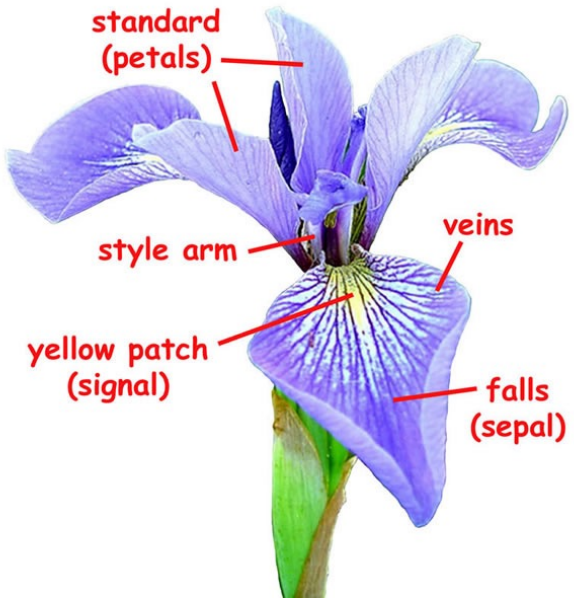
SECTION 3

Case Studies

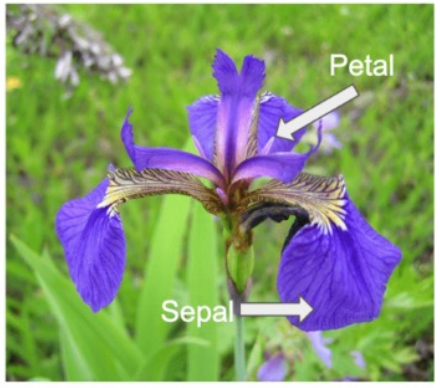


Case Study 1

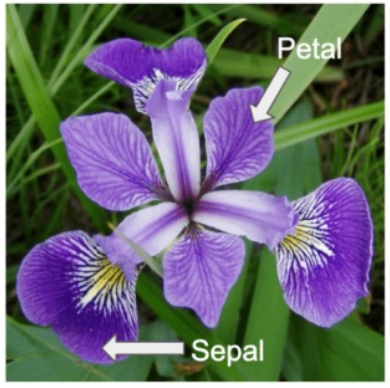
❖ IRIS Flower



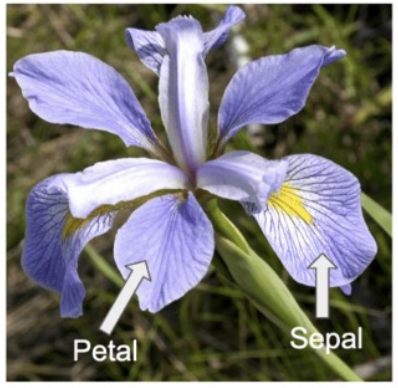
Iris setosa



Iris versicolor



Iris virginica



	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3.0	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5.0	3.6	1.4	0.2	Setosa
5	5.4	3.9	1.7	0.4	Setosa
6	4.6	3.4	1.4	0.3	Setosa
7	5.0	3.4	1.5	0.2	Setosa
8	4.4	2.9	1.4	0.2	Setosa
9	4.9	3.1	1.5	0.1	Setosa
10	5.4	3.7	1.5	0.2	Setosa
11	4.8	3.4	1.6	0.2	Setosa
12	4.8	3.0	1.4	0.1	Setosa
13	4.3	3.0	1.1	0.1	Setosa
14	5.8	4.0	1.2	0.2	Setosa
15	5.7	4.4	1.5	0.4	Setosa
16	5.4	3.9	1.3	0.4	Setosa
17	5.1	3.5	1.4	0.3	Setosa
18	5.7	3.8	1.7	0.3	Setosa
19	5.1	3.8	1.5	0.3	Setosa

Case Study 1

❖ IRIS Flower

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import seaborn as sns
4
5  data = pd.read_csv('iris.csv')
```

data.head()

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	Setosa
1	4.9	3	1.4	0.2	Setosa
2	4.7	3.2	1.3	0.2	Setosa
3	4.6	3.1	1.5	0.2	Setosa
4	5	3.6	1.4	0.2	Setosa

data.describe()

	sepal_length	sepal_width	petal_length	petal_width
count	150	150	150	150
mean	5.843	3.057	3.758	1.199
std	0.828	0.435	1.765	0.762
min	4.3	2	1	0.1
0.25	5.1	2.8	1.6	0.3
0.5	5.8	3	4.35	1.3
0.75	6.4	3.3	5.1	1.8
max	7.9	4.4	6.9	2.5

data.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal.length  150 non-null    float64
 1   sepal.width   150 non-null    float64
 2   petal.length  150 non-null    float64
 3   petal.width   150 non-null    float64
 4   Species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

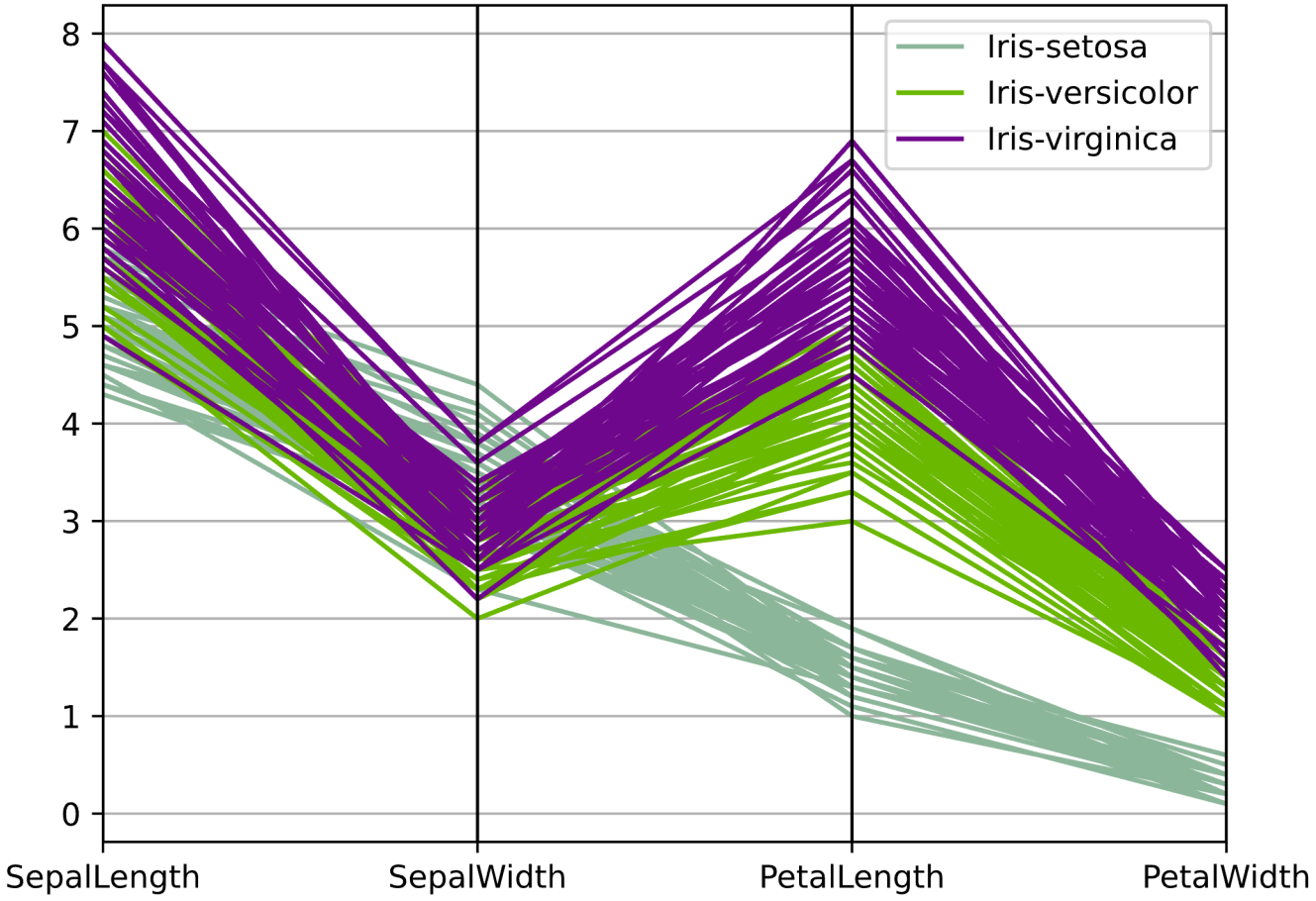
data["species"].value_counts()

```
Setosa        50
Versicolor    50
Virginica     50
Name: species, dtype: int64
```

Case Study 1

❖ IRIS Flower

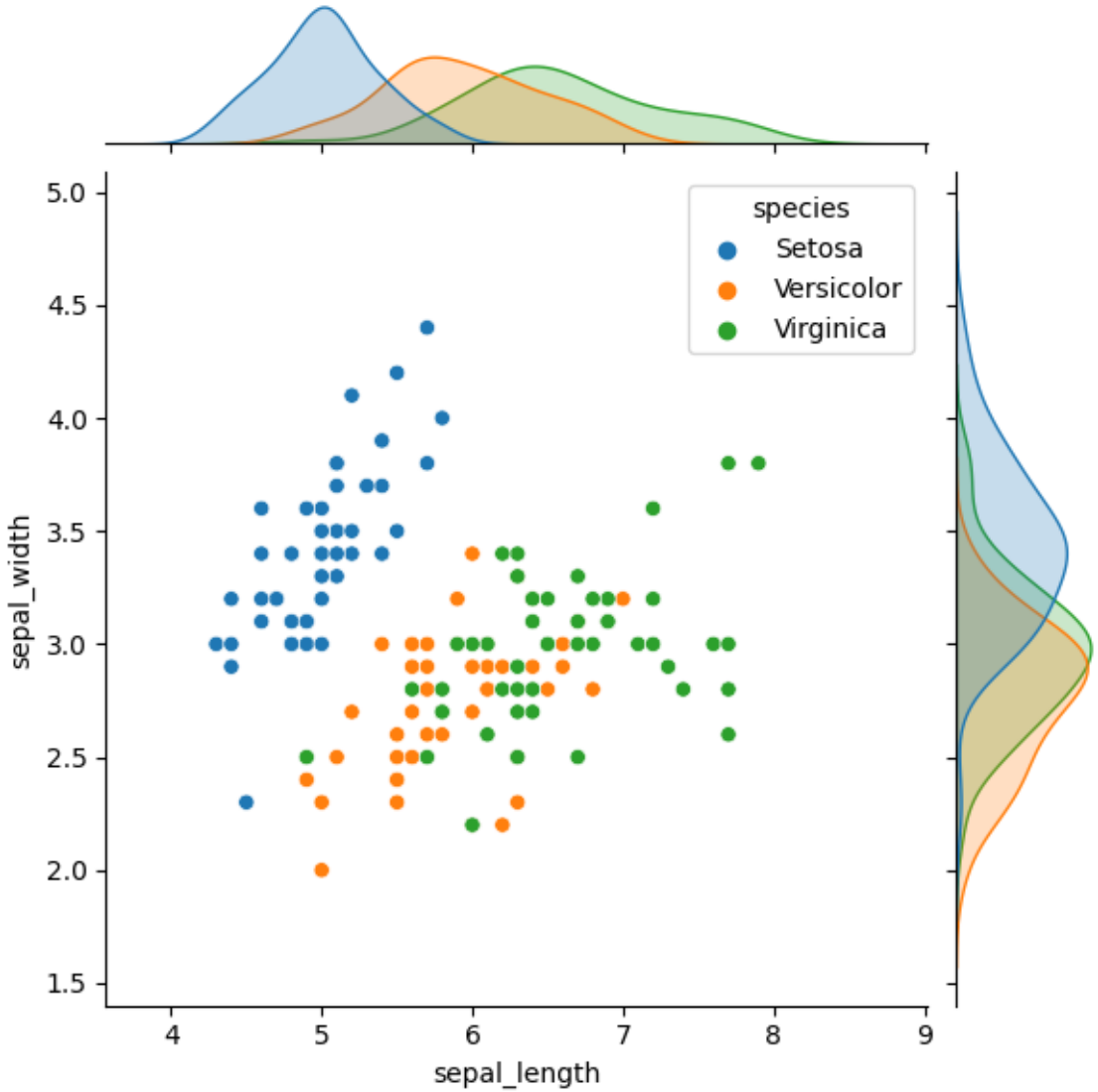
```
data = pd.read_csv("iris.csv")
pd.plotting.parallel_coordinates(data, "Name")
```



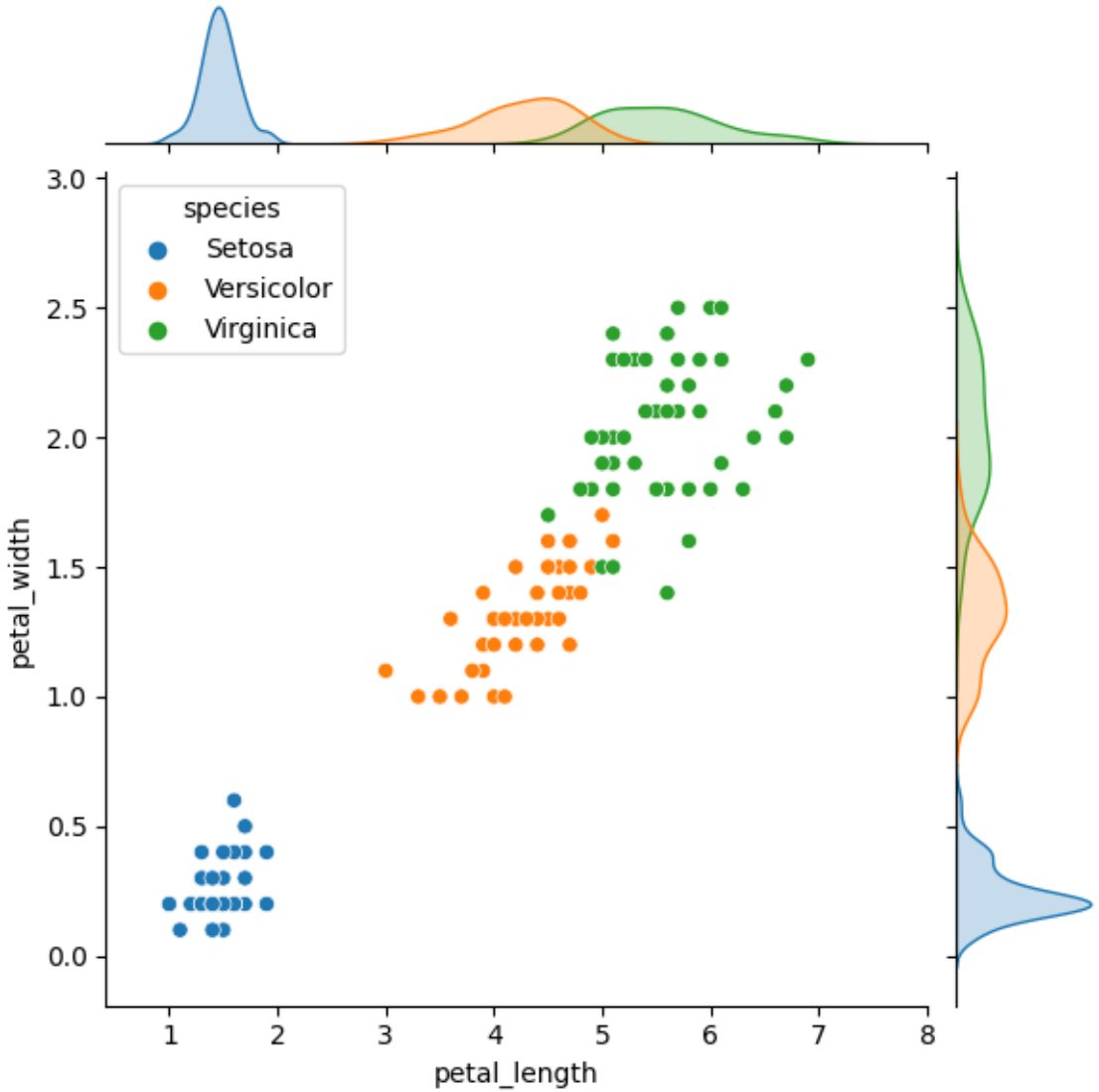
SepalLength	SepalWidth	PetalLength	PetalWidth	Name
5.1	3.5	1.4	0.2	Iris-setosa
4.9	3.0	1.4	0.2	Iris-setosa
4.7	3.2	1.3	0.2	Iris-setosa
4.6	3.1	1.5	0.2	Iris-setosa
5.0	3.6	1.4	0.2	Iris-setosa
5.4	3.9	1.7	0.4	Iris-setosa
4.6	3.4	1.4	0.3	Iris-setosa
5.0	3.4	1.5	0.2	Iris-setosa
4.4	2.9	1.4	0.2	Iris-setosa
4.9	3.1	1.5	0.1	Iris-setosa
5.4	3.7	1.5	0.2	Iris-setosa
4.8	3.4	1.6	0.2	Iris-setosa
4.8	3.0	1.4	0.1	Iris-setosa
4.3	3.0	1.1	0.1	Iris-setosa
5.8	4.0	1.2	0.2	Iris-setosa
5.7	4.4	1.5	0.4	Iris-setosa
5.4	3.9	1.3	0.4	Iris-setosa
5.1	3.5	1.4	0.3	Iris-setosa
5.7	3.8	1.7	0.3	Iris-setosa
5.1	3.8	1.5	0.3	Iris-setosa
5.4	3.4	1.7	0.2	Iris-setosa
5.1	3.7	1.5	0.4	Iris-setosa
4.6	3.6	1.0	0.2	Iris-setosa
5.1	3.3	1.7	0.5	Iris-setosa

Case study 1

```
sns.jointplot(x="sepal_length", y="sepal_width",
              data=data, hue="species")
plt.show()
```

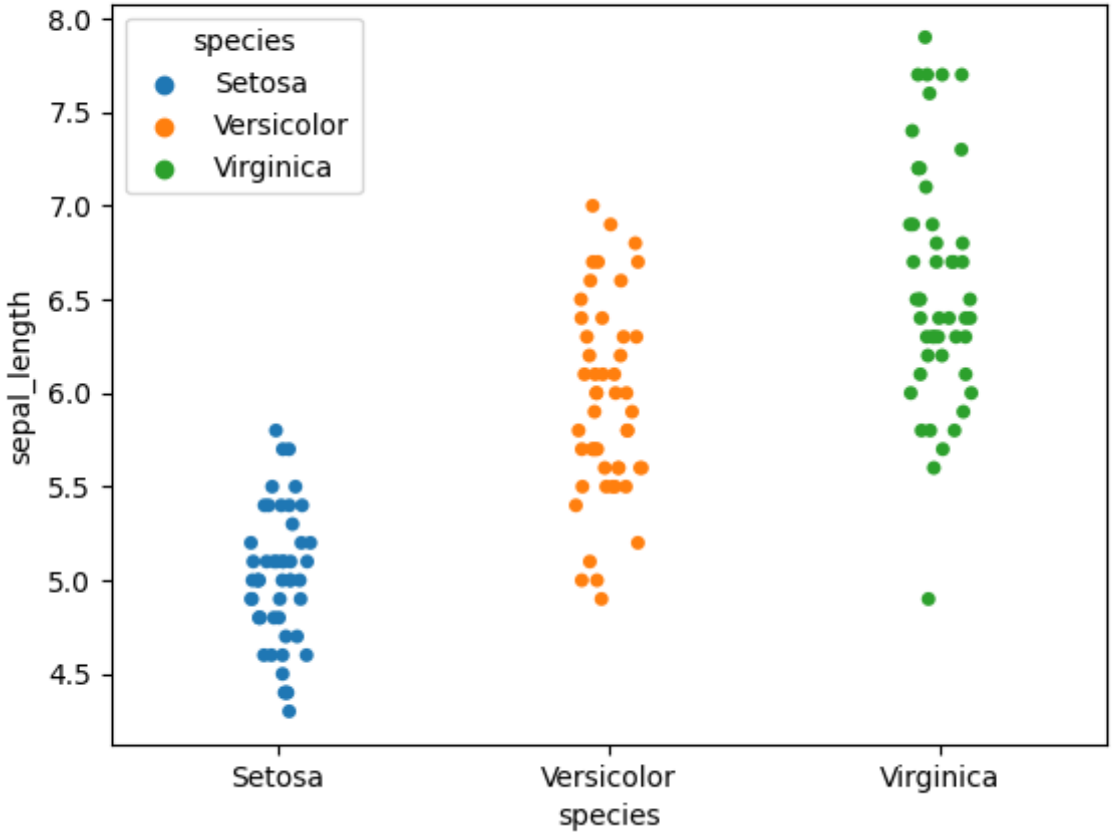



```
sns.jointplot(x="petal_length", y="sepal_width",
              data=data, hue="species")
plt.show()
```

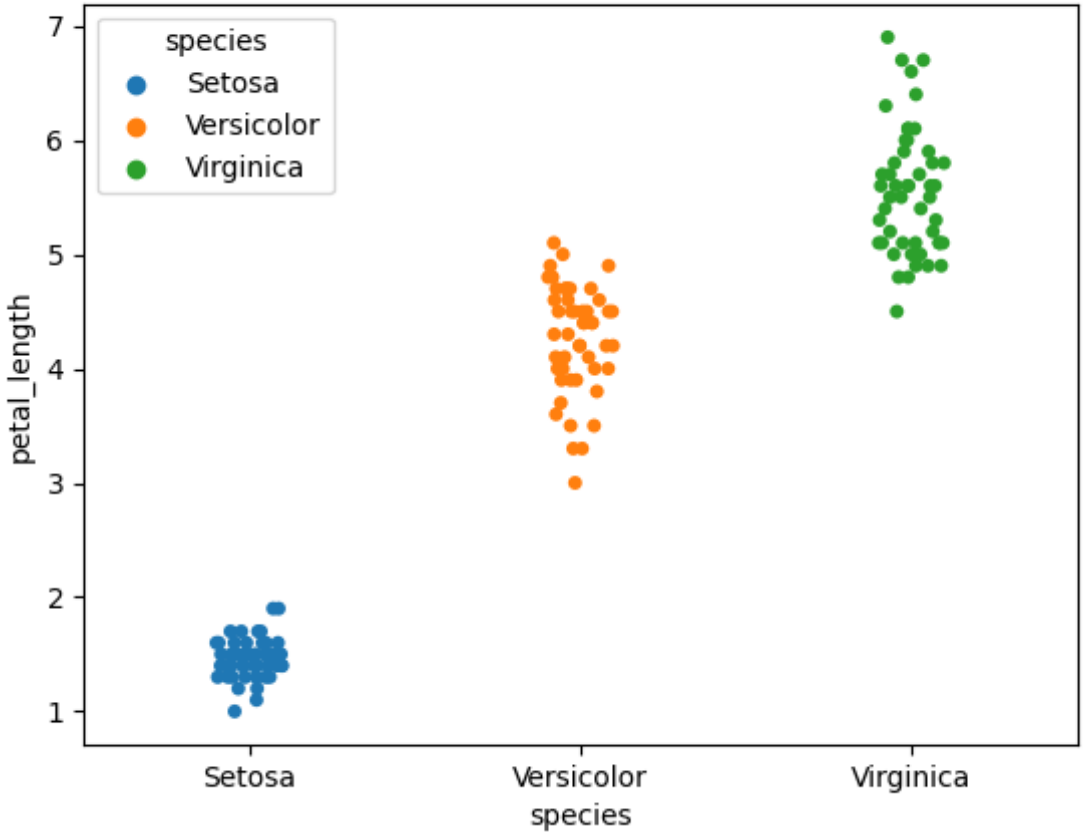


Case study 1

```
sns.stripplot(y='sepal_length', x='species',
              data=data, hue="species")
plt.show()
```

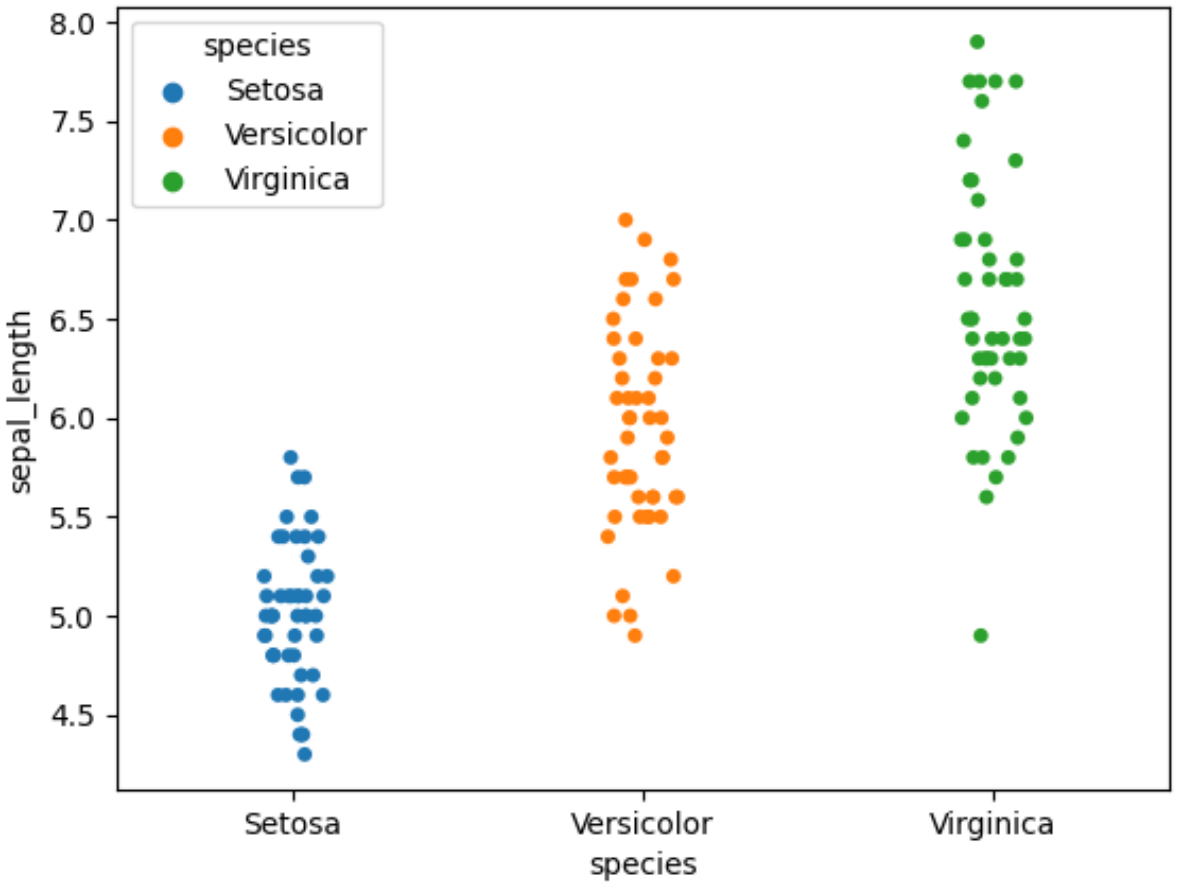



```
sns.stripplot(y='petal_length', x='species',
              data=data, hue="species")
plt.show()
```

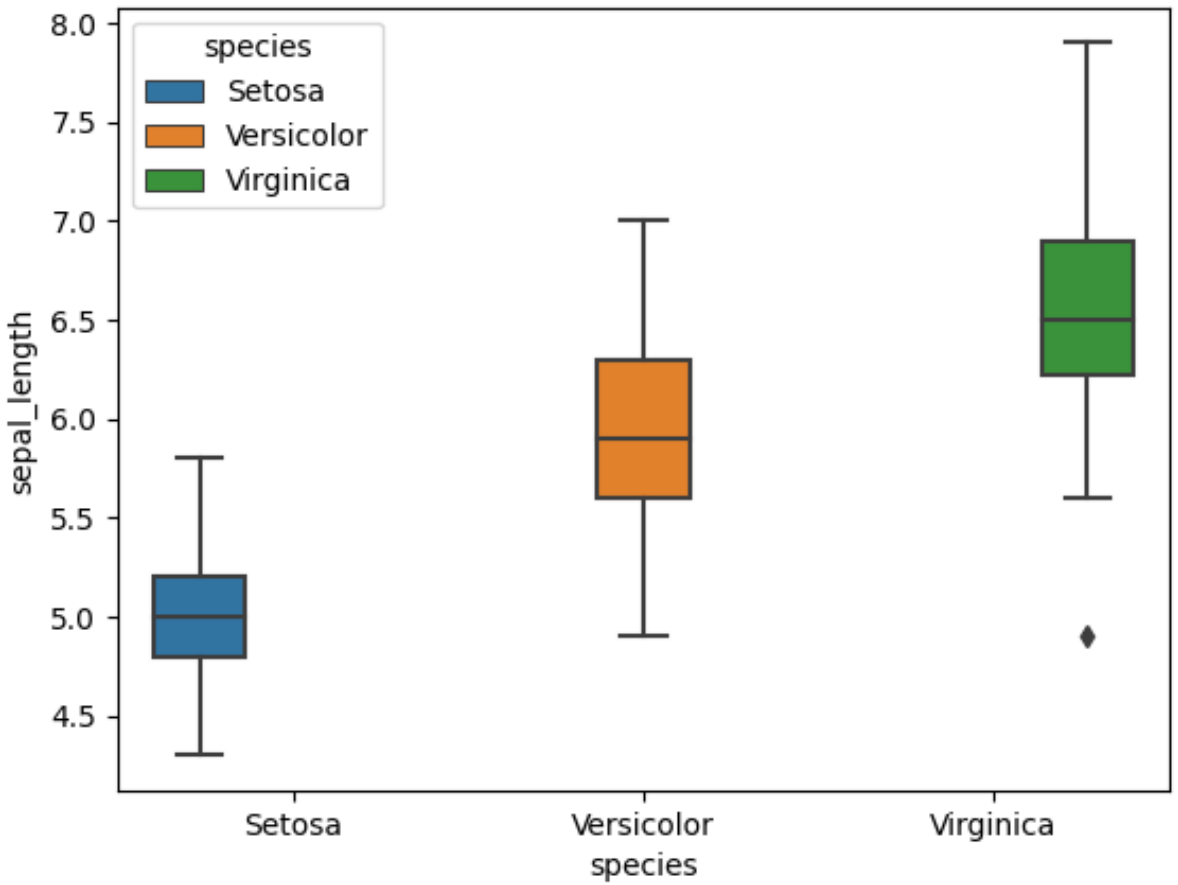


Case study 1

```
sns.stripplot(y='sepal_length', x='species',
              data=data, hue="species")
plt.show()
```

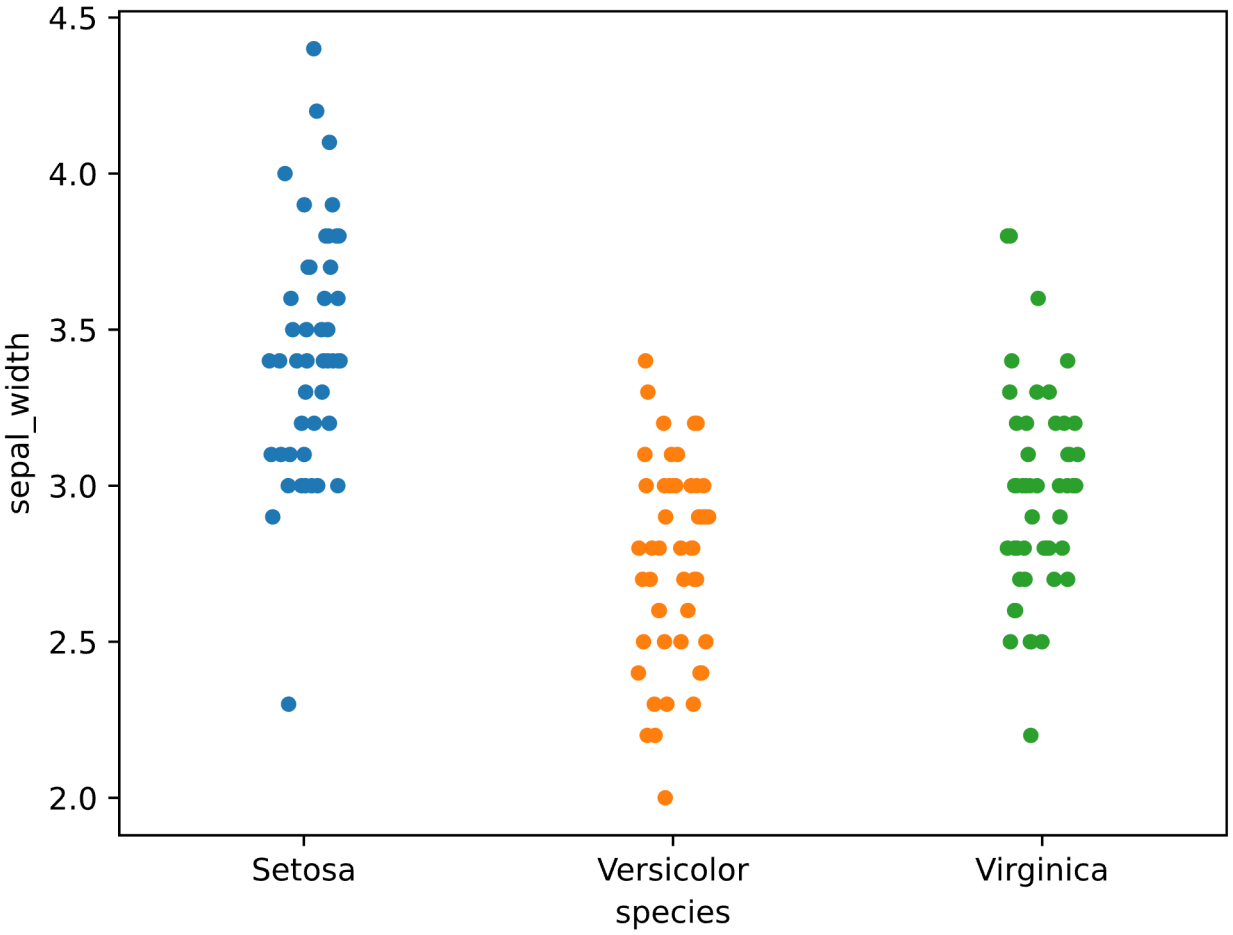


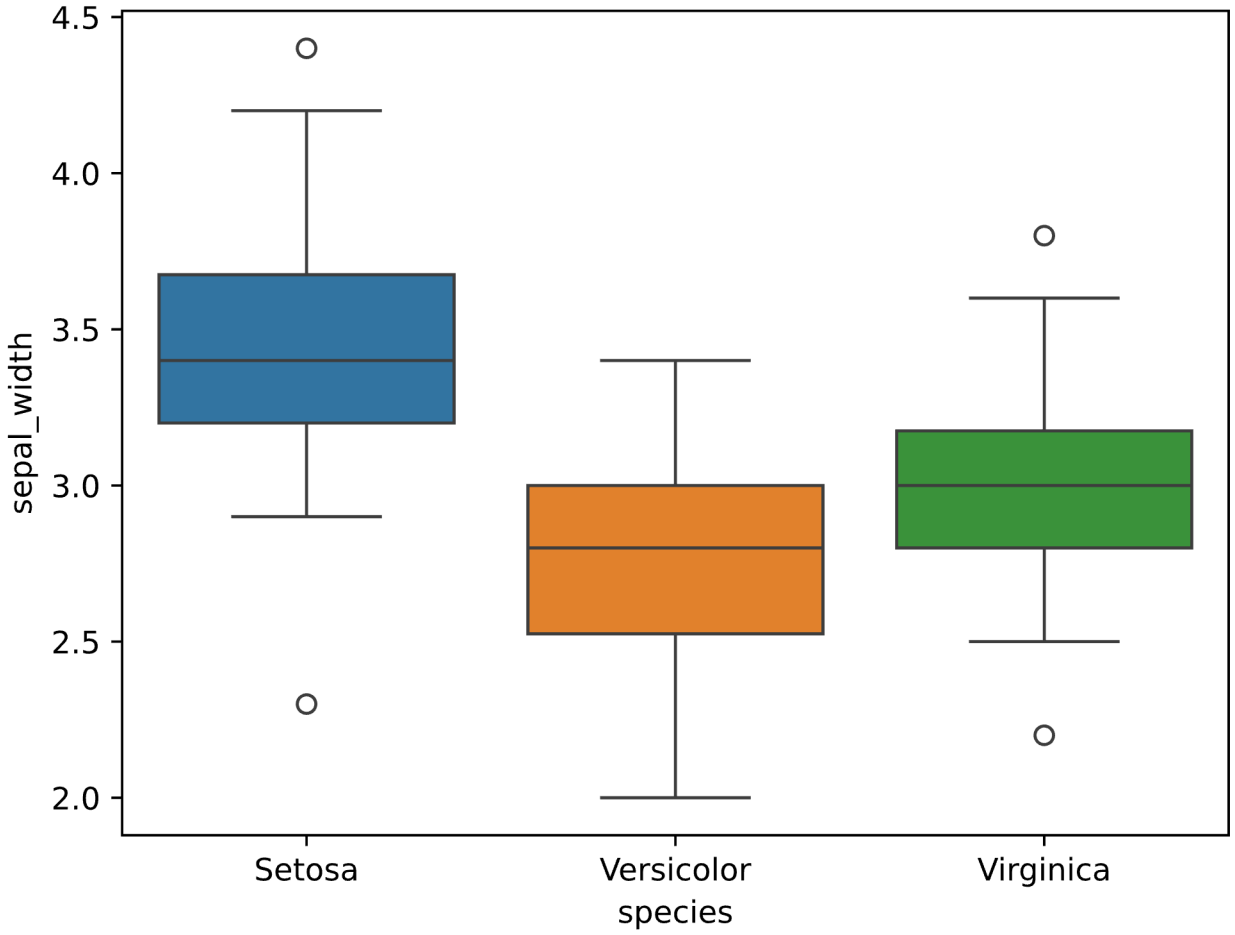

```
sns.boxplot(x="species", y="sepal_length",
            data=data, hue="species")
plt.show()
```



Case study 1: Quiz

Estimate the boxplot for each category





Correlation Coefficient

population

❖ Definition

Công thức: Gọi x,y là hai biến ngẫu nhiên

$$\rho_{xy} = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n(\sum_i x_i y_i) - (\sum_i x_i)(\sum_i y_i)}{\sqrt{n\sum_i x_i^2 - (\sum_i x_i)^2}\sqrt{n\sum_i y_i^2 - (\sum_i y_i)^2}}$$

Tính chất 1

$$-1 \leq \rho_{xy} \leq 1$$

Tương quan nghịch ← | → Tương quan thuận

Tính chất 2

$$\rho_{xy} = \rho_{uv}$$

trong đó

$$u = ax + b$$

$$v = cy + d$$

Ví dụ 1

x = [7, 18, 29, 2, 10, 9, 9]

y = [1, 6, 12, 8, 6, 21, 10]

$$\rho_{xy} = \frac{E[(x-\mu_x)(y-\mu_y)]}{\sqrt{var(x)}\sqrt{var(y)}}$$

$$= \frac{n*818 - 84*64}{\sqrt{n*1480 - 7056}\sqrt{n*822 - 4096}} = 0.149$$

Ví dụ 2

u=2*x-14 = [0, 22, 44, -10, 6, 4, 4]

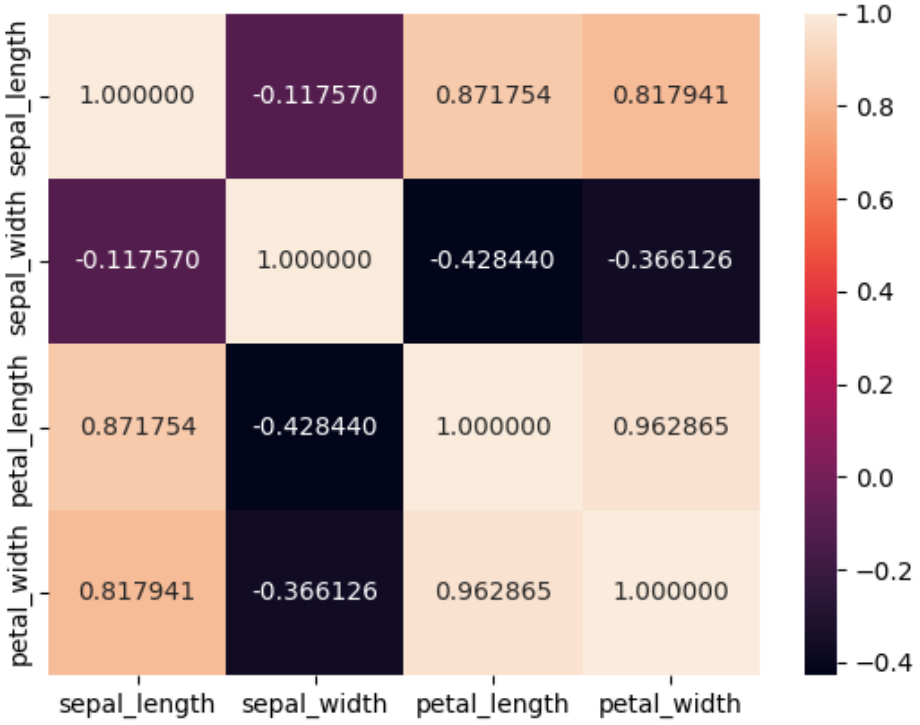
v=y+2 = [3, 8, 14, 10, 8, 23, 12]

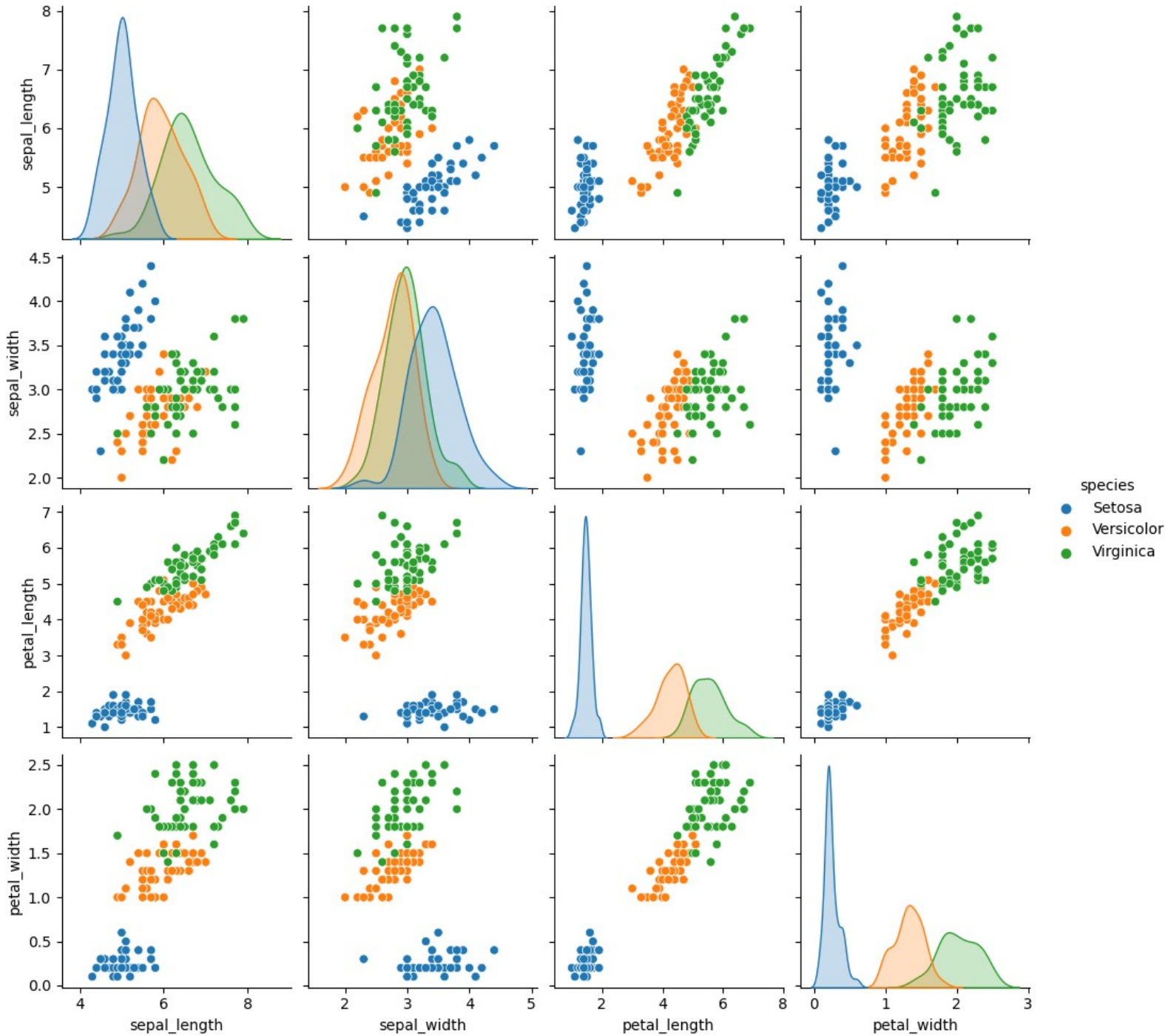
$$\rho_{uv} = \frac{E[(u-\mu_u)(v-\mu_v)]}{\sqrt{var(u)}\sqrt{var(v)}}$$

$$= \frac{n*880 - 70*78}{\sqrt{n*2588 - 4900}\sqrt{n*1106 - 6084}} = 0.149$$

Case study 1

```
sns.pairplot(data=data,
                hue="species")
sns.heatmap(data.iloc[:,0:4].corr(),
                annot=True, fmt="f")
plt.show()
```





Outline

SECTION 1

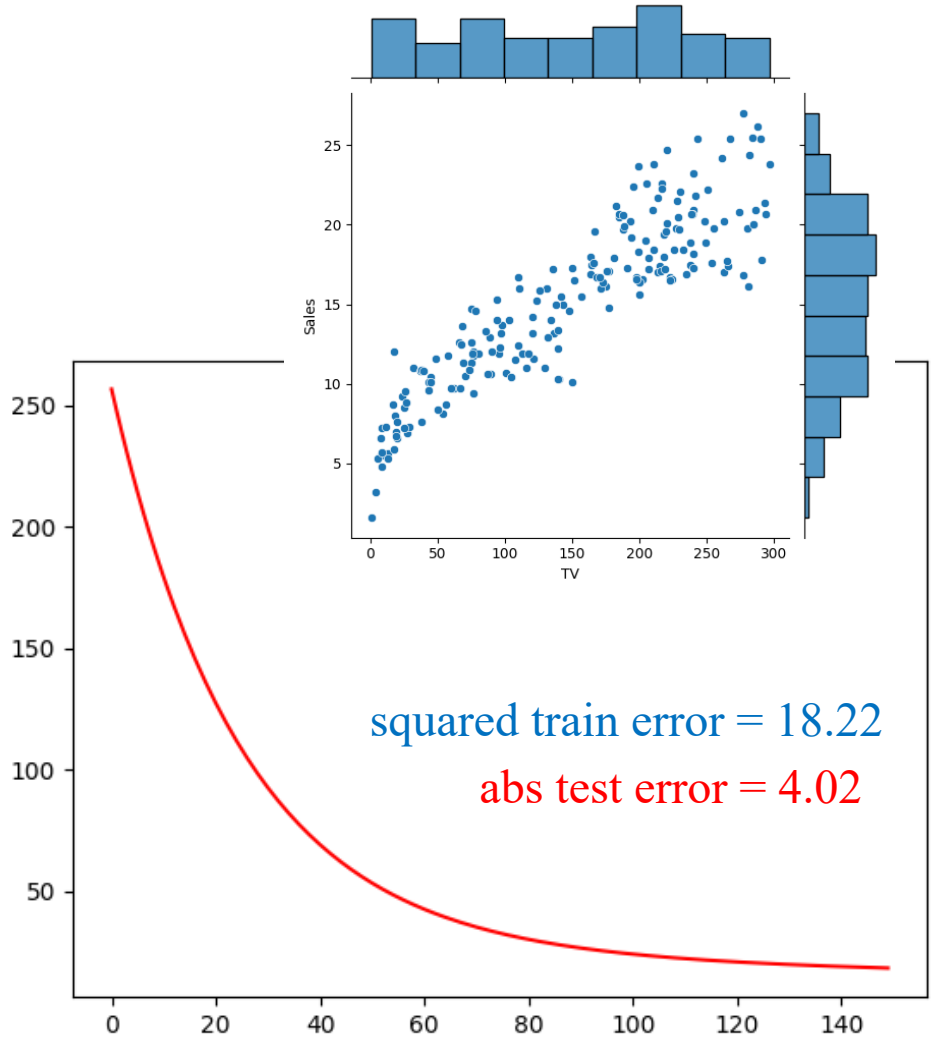
DataFrame in Pandas

SECTION 2

Visualization

SECTION 3

Case Studies



Case study 2

House price data

Feature	Label
area	price
6.7	9.1
4.6	5.9
3.5	4.6
5.5	6.7

price = a $*$ $area + b$

Advertising data

Features			Label
TV	Radio	Newspaper	Sales
230.1	37.8	69.2	22.1
44.5	39.3	45.1	10.4
17.2	45.9	69.3	12
151.5	41.3	58.5	16.5
180.8	10.8	58.4	17.9

Model: $\hat{y} = w_1x_1 + w_2x_2 + w_3x_3 + b$

Sale = $w_1 * TV + w_2 * Radio + w_3 * Newspaper + b$

Boston House Price Data

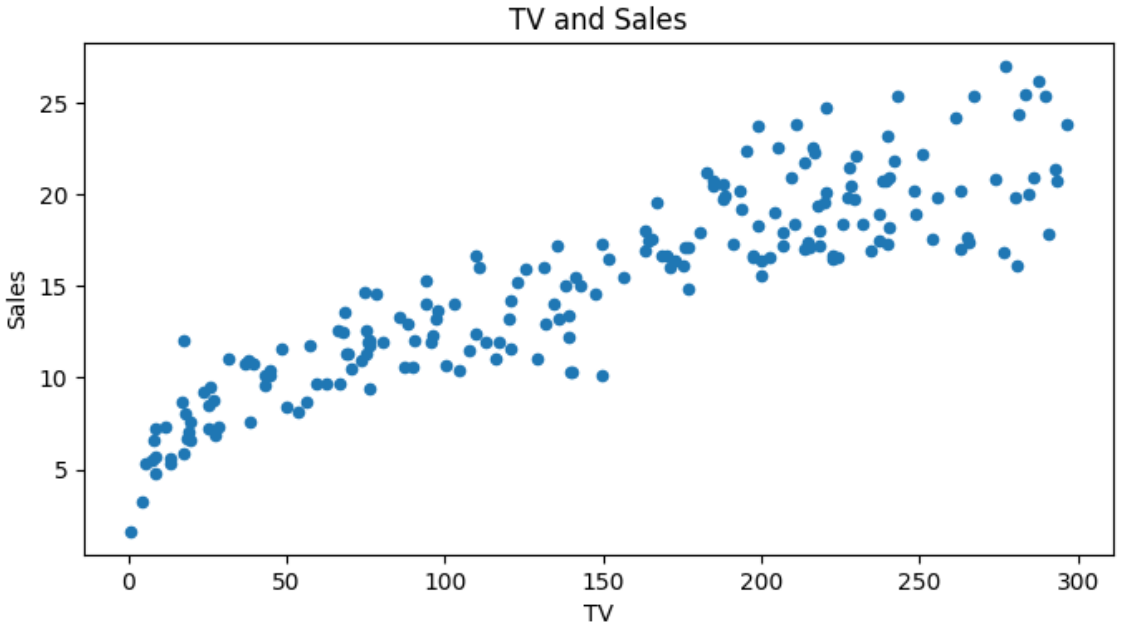
Features													Label
crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	black	lstat	medv
0.00632	18	2.31	0	0.538	6.575	65.2	4.09	1	296	15.3	396.9	4.98	24
0.02731	0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.9	9.14	21.6
0.03237	0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94	33.4
0.06905	0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.9	5.33	36.2
0.08829	12.5	7.87	0	0.524	6.012	66.6	5.5605	5	311	15.2	395.6	12.43	22.9

medv = $w_1 * x_1 + \cdots + w_{13} * x_{13} + b$

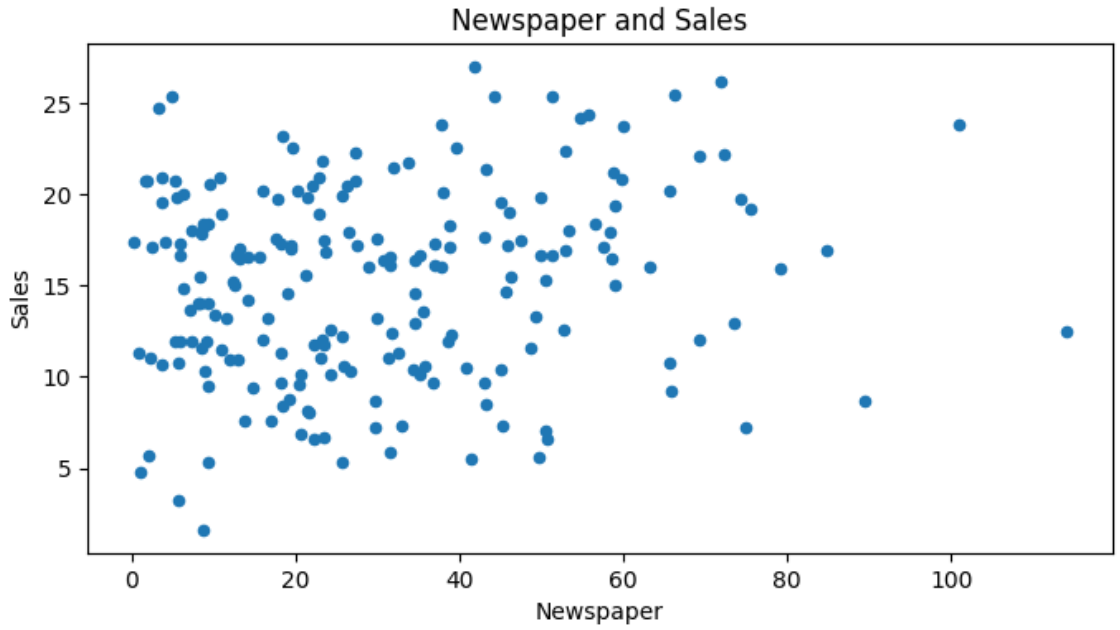
Case study 2

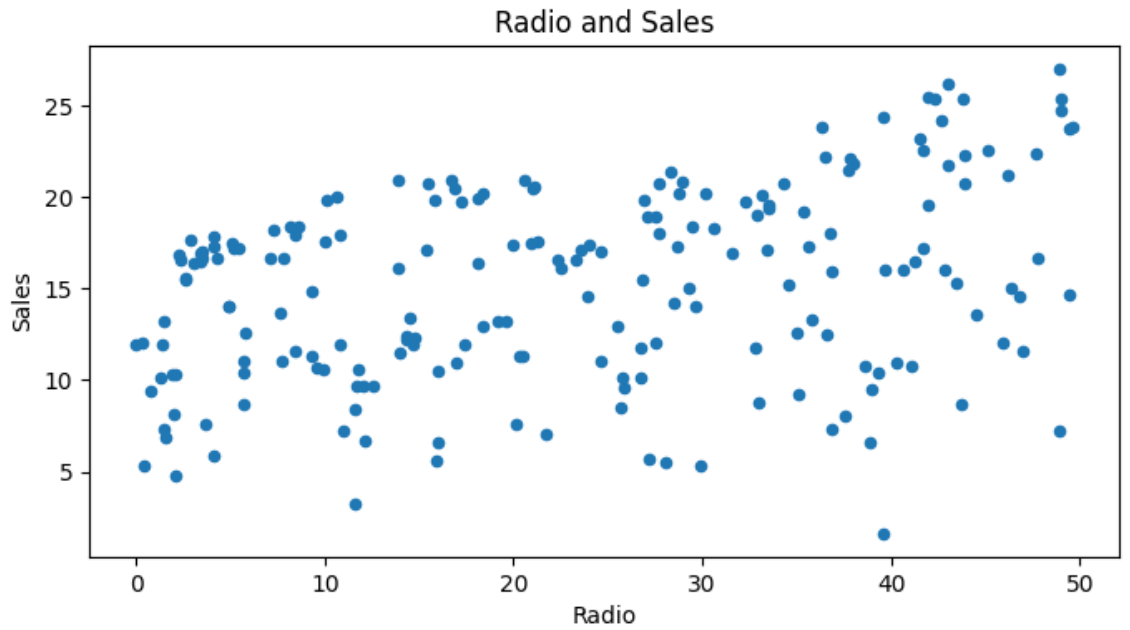
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...

```
1  import pandas as pd
2
3  df = pd.read_csv('advertising.csv')
4
5  df.plot(x='TV', y='Sales', kind='scatter')
6  df.plot(x='Radio', y='Sales', kind='scatter')
7  df.plot(x='Newspaper', y='Sales', kind='scatter')
```



2.1.CSV_Reading.ipynb





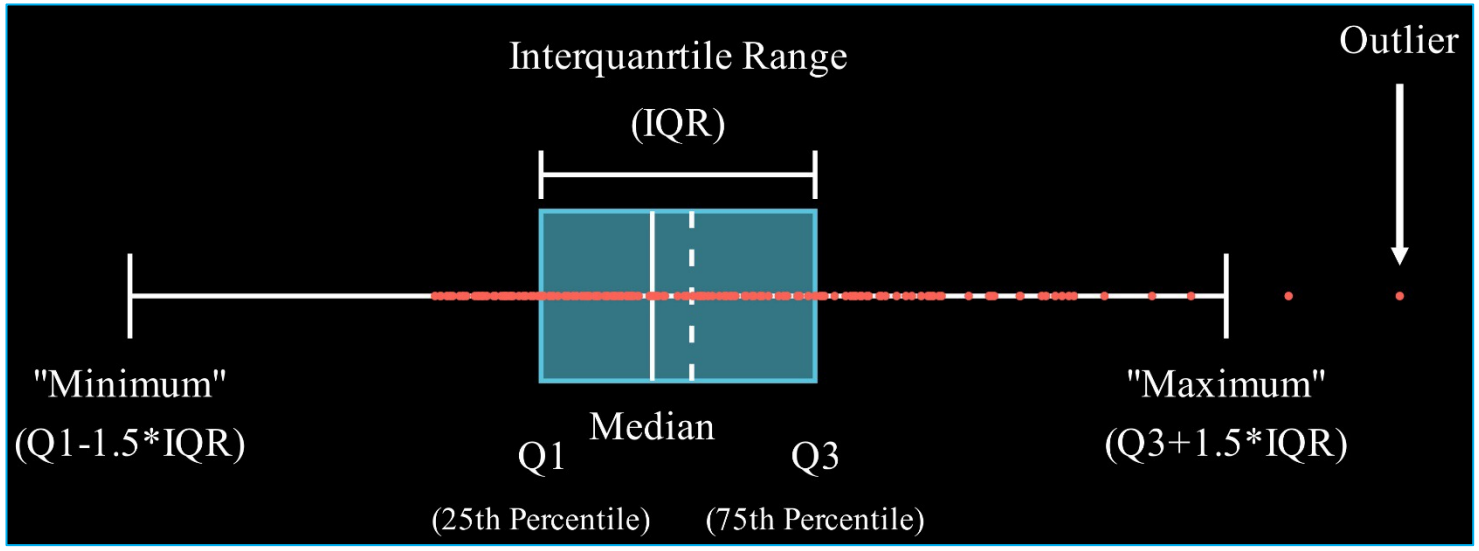
Case Study 2

❖ Box plot

```
1  import numpy as np
2  import pandas as pd
3
4  df = pd.read_csv('advertising.csv')
5  df.describe()
```

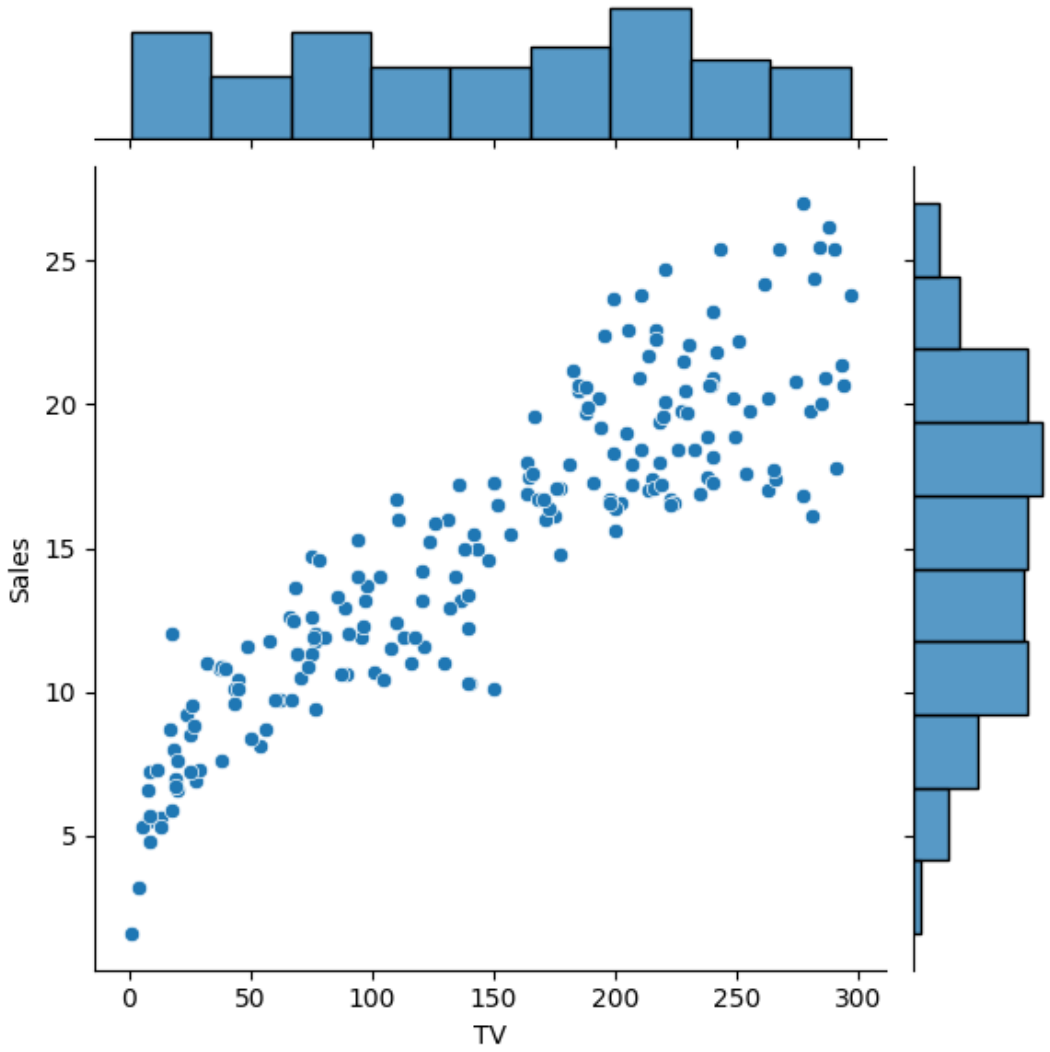
	TV	Radio	Newspaper	Sales
count	200.000000	200.000000	200.000000	200.000000
mean	147.042500	23.264000	30.554000	15.130500
std	85.854236	14.846809	21.778621	5.283892
min	0.700000	0.000000	0.300000	1.600000
25%	74.375000	9.975000	12.750000	11.000000
50%	149.750000	22.900000	25.750000	16.000000
75%	218.825000	36.525000	45.100000	19.050000
max	296.400000	49.600000	114.000000	27.000000

	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...



Case Study 2

❖ Scatter data



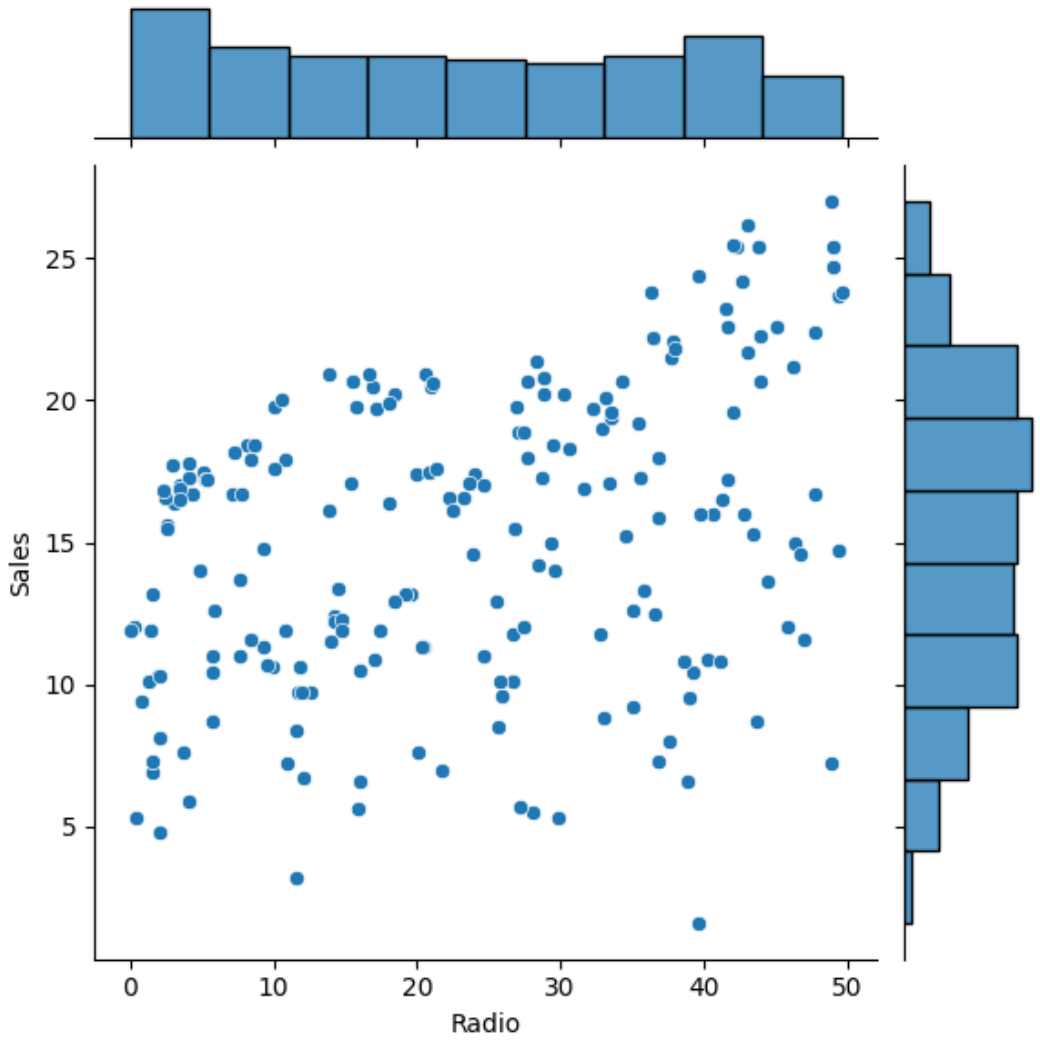
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...

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

data = pd.read_csv('advertising.csv')
sns.jointplot(x="TV",
              y="Sales",
              data=data)
plt.show()
```

Case Study 2

❖ Scatter data

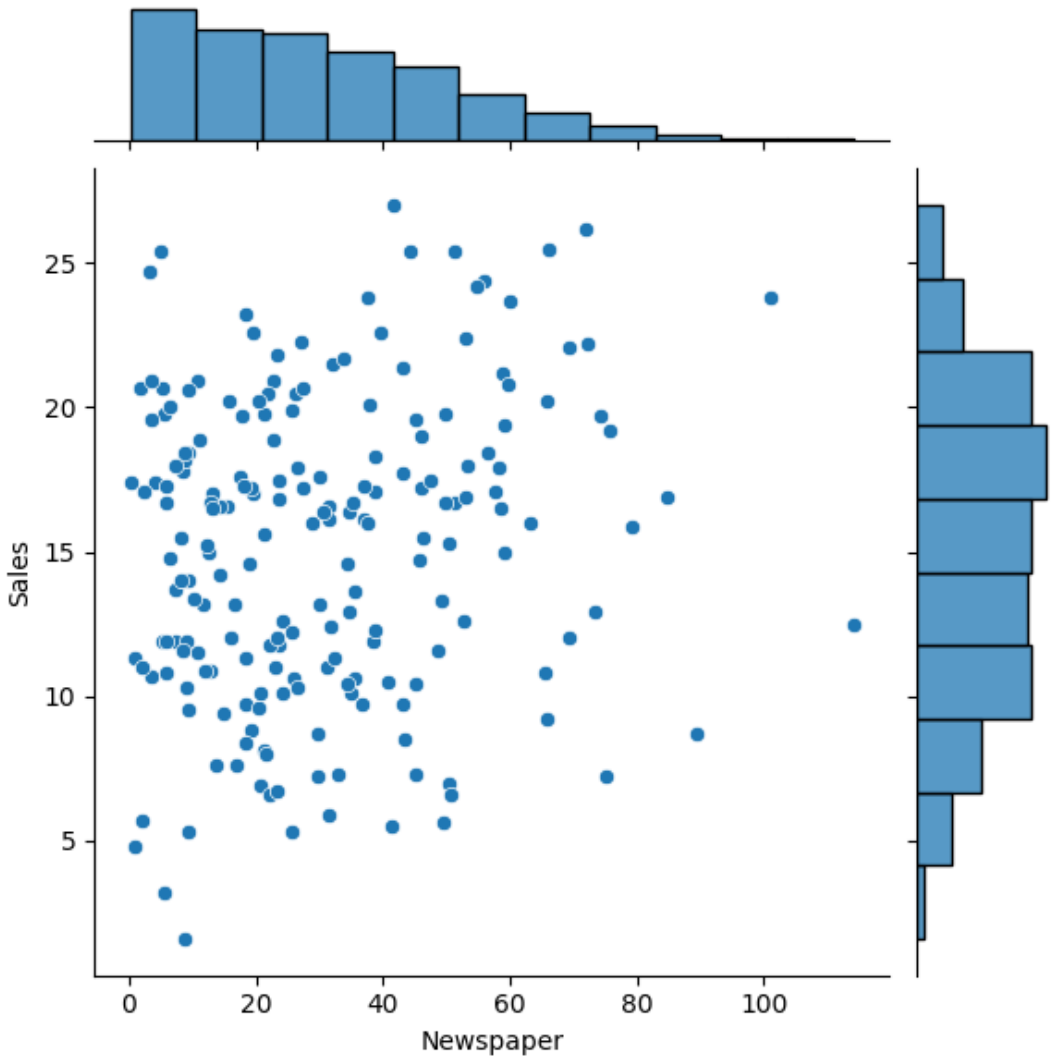


	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import seaborn as sns
4
5  data = pd.read_csv('advertising.csv')
6  sns.jointplot(x="Radio",
7                y="Sales",
8                data=data)
9  plt.show()
```

Case Study 2

❖ Scatter data



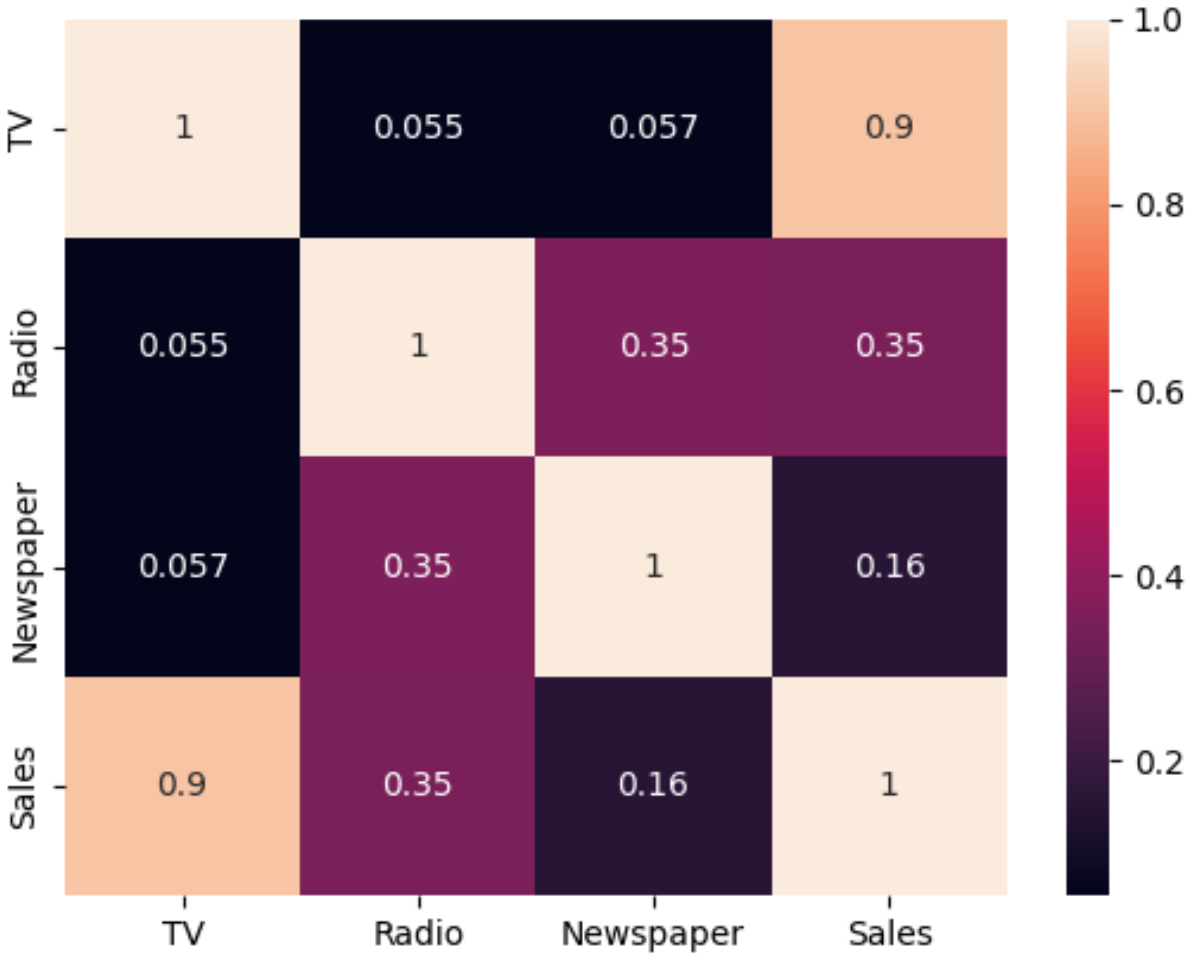
	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...

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

data = pd.read_csv('advertising.csv')
sns.jointplot(x="Newspaper",
              y="Sales",
              data=data)
plt.show()
```

Case Study 2

❖ Correlation



	TV	Radio	Newspaper	Sales
0	230.1	37.8	69.2	22.1
1	44.5	39.3	45.1	10.4
2	17.2	45.9	69.3	12
3	151.5	41.3	58.5	16.5
4	180.8	10.8	58.4	17.9
...	...	...	...	...

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import seaborn as sns
4
5  sns.heatmap(data.corr(),
6              annot=True)
7  plt.show()
```

Case study 2

```python
import numpy as np

epochs = 150
lr = 0.01

# khởi tạo giá trị tham số
thetas = np.random.randn(F, 1)

for i in range(epochs):
    # tính output
    output = x_train.dot(thetas)

    # tính loss
    loss = (output - y_train)**2

    # tính đạo hàm cho loss
    loss_grd = 2*(output - y_train)/N

    # tính đạo hàm cho các tham số
    gradients = x_train.T.dot(loss_grd)

    # cập nhật tham số
    thetas = thetas - lr*gradients

# compute abs test error
output = x_test.dot(thetas)
test_loss = np.abs(output - y_test)
```