Preparing Machine Learning

Data Preprocessing

Using pandas, matplotlib, scikit-learn

경희대학교 컴퓨터공학과
2019102191 신주영

경희대학교
KYUNG HEE UNIVERSITY

Importance of Collecting and Preparing Data

- To solve real problems with machine learning, data must be collected.
- After collecting data, data preprocessing is required, such as setting the correct answer, modifying it in a form that is easy to learn, deleting unnecessary data, or adding other data.
- Data preprocessing is such an important task that it accounts for 80% of the entire machine learning process.

Dependent variable and Independent variable

- Independent variable: The data based on prediction
- Dependent variable: The data to be predicted
- Independent variables are also called feature or input variable
- Dependent variables are also called label or class label data

Looking for scikit-learn's sample data

get iris data set

- Import pandas
- Get iris datasets
- Make data frame using pandas
- Combine target data and feature data

```
import pandas as pd
from sklearn.datasets import load_iris

data = load_iris()

X = pd.DataFrame(data.data, columns = data.feature_names)
y = pd.DataFrame(data.target, columns = ['Species'])

df = pd.concat([X, y], axis = 1)
df.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	Species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

Example of supervised learning

looking for example (breast cancer diagnosis data set, data preprocessing)

- Import scikit-learn datasets
- Get data from datasets
- Set dependent variable and independent variable
- Get only the data we want

```
from sklearn.datasets import load_breast_cancer

data = load_breast_cancer()

X = data.data
y = data.target

X = X[:, :10]
```

Example of supervised learning

looking for example (breast cancer diagnosis data set, data preprocessing)

	radius mean	texture mean	perimeter mean	area mean	smoothness mean	compactness mean	concavity mean	concave points mean	symmetry mean	fractal dimension mean	type
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	0
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	0
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	0
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	0
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	0

	radius error	texture error	perimeter error	area error	smoothness error	compactness error	concavity error	concave points error	symmetry error	fractal dimension error	type
0	1.0950	0.9053	8.589	153.40	0.006399	0.04904	0.05373	0.01587	0.03003	0.006193	0
1	0.5435	0.7339	3.398	74.08	0.005225	0.01308	0.01860	0.01340	0.01389	0.003532	0
2	0.7456	0.7869	4.585	94.03	0.006150	0.04006	0.03832	0.02058	0.02250	0.004571	0
3	0.4956	1.1560	3.445	27.23	0.009110	0.07458	0.05661	0.01867	0.05963	0.009208	0
4	0.7572	0.7813	5.438	94.44	0.011490	0.02461	0.05688	0.01885	0.01756	0.005115	0

	radius worst	texture worst	perimeter worst	area worst	smoothness worst	compactness worst	concavity worst	concave points worst	symmetry worst	fractal dimension worst	type
0	25.38	17.33	184.60	2019.0	0.1622	0.6656	0.7119	0.2654	0.4601	0.11890	0
1	24.99	23.41	158.80	1956.0	0.1238	0.1866	0.2416	0.1860	0.2750	0.08902	0
2	23.57	25.53	152.50	1709.0	0.1444	0.4245	0.4504	0.2430	0.3613	0.08758	0
3	14.91	26.50	98.87	567.7	0.2098	0.8663	0.6869	0.2575	0.6638	0.17300	0
4	22.54	16.67	152.20	1575.0	0.1374	0.2050	0.4000	0.1625	0.2364	0.07678	0

경희대학교
KYUNG HEE UNIVERSITY

Example of supervised learning

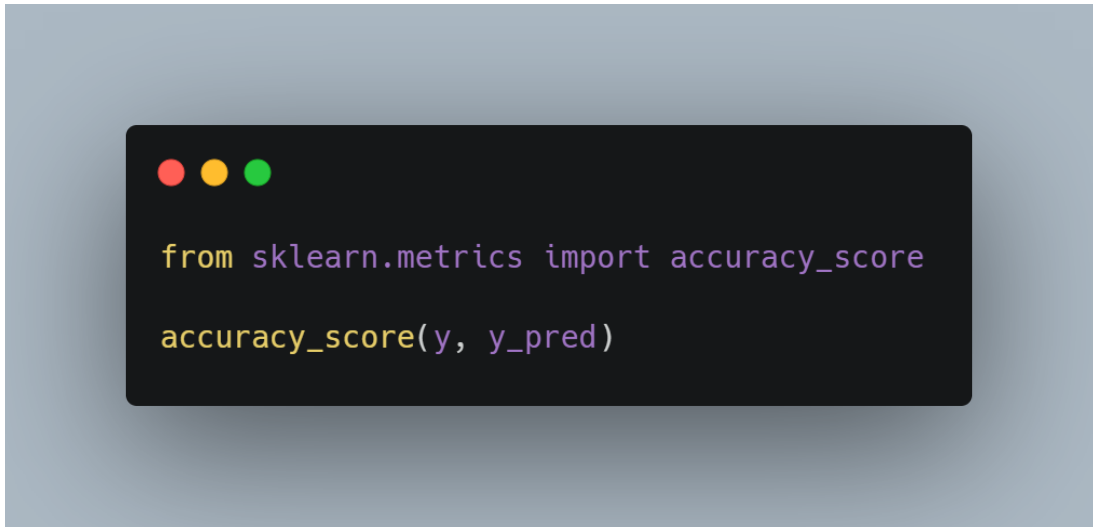
logistic regression

- Import logistic regression from scikit-learn
- Make instance of logistic regression class
- Learn by executing the fit method in the model variable
- The prediction is made by executing the predict method on the learned model

```
from sklearn.linear_model import LogisticRegression

model = LogisticRegression(solver = 'lbfgs')

model.fit(X, y)

y_pred = model.predict(X)
```

Accuracy evaluation

- To calculate accuracy, use the scikit-learn's accuracy_score function
- The accuracy is output by entering the dependent variable y and the y_pred in the accuracy_score function



Out [9] : 0.9121265377855887

Example of unsupervised learning

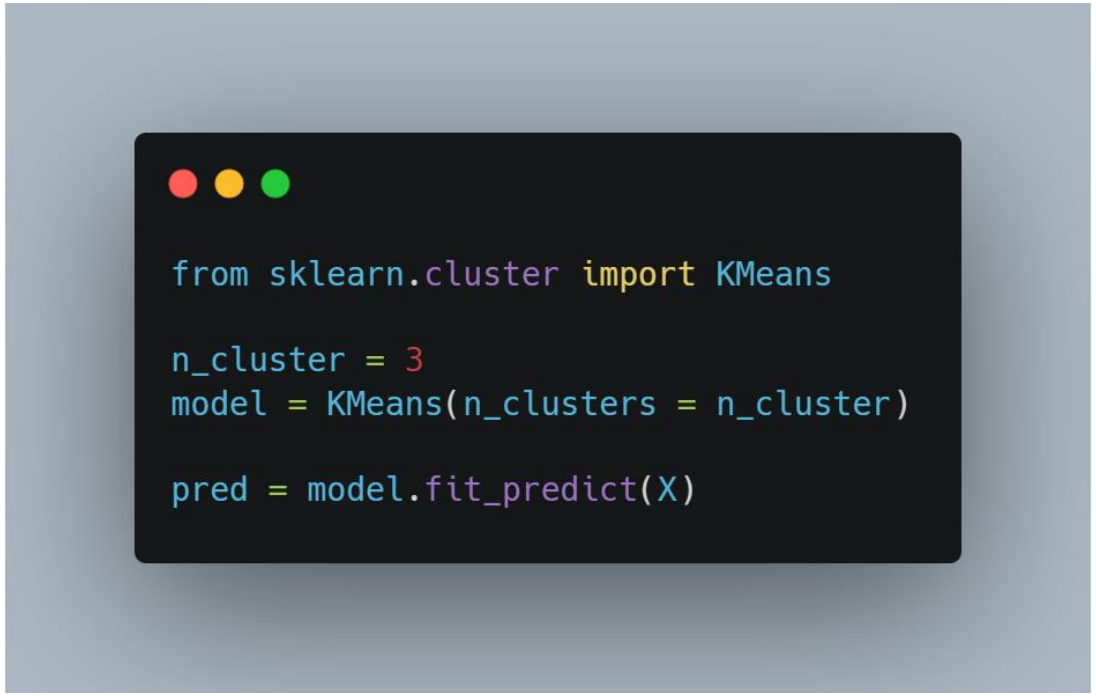
looking for example (wine data set, data preprocessing)

- Import scikit-learn datasets
- Get data from datasets
- Set only dependent variable
- Get only the data we want

Example of unsupervised learning

K-Means algorithm

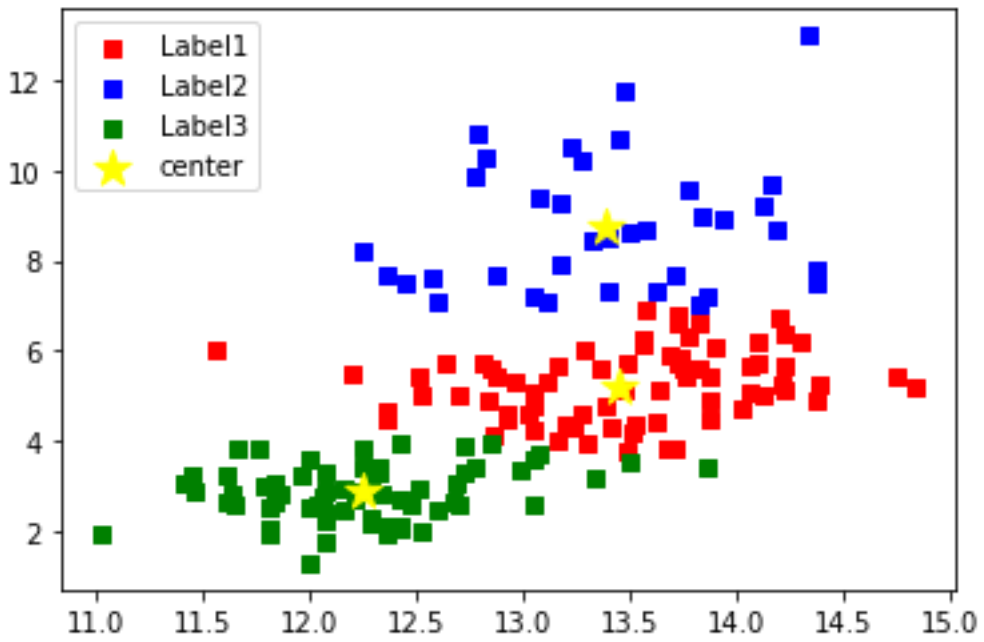
- Import K-Means algorithm from scikit-learn
- We will divide the data into three groups.(n_cluster = 3)
- By using fit_predict method, we can do learning and prediction.



```
from sklearn.cluster import KMeans

n_cluster = 3
model = KMeans(n_clusters = n_cluster)

pred = model.fit_predict(X)
```

Confirmation of learning output

- By using visualization, we're going to confirm clustering results.




```
%matplotlib inline
import matplotlib.pyplot as plt

fig, ax = plt.subplots()

ax.scatter(X[pred==0, 0], X[pred == 0, 1], color = 'red', marker = 's', label =
'Label1')
ax.scatter(X[pred==1, 0], X[pred == 1, 1], color = 'blue', marker = 's', label =
'Label2')
ax.scatter(X[pred==2, 0], X[pred == 2, 1], color = 'green', marker = 's', label =
'Label3')
ax.scatter(model.cluster_centers_[:, 0], model.cluster_centers_[:, 1], s = 200, color
= 'yellow', marker = '*', label = 'center')
ax.legend()

plt.show()
```

Visualization

using python (pandas, matplotlib, seaborn)

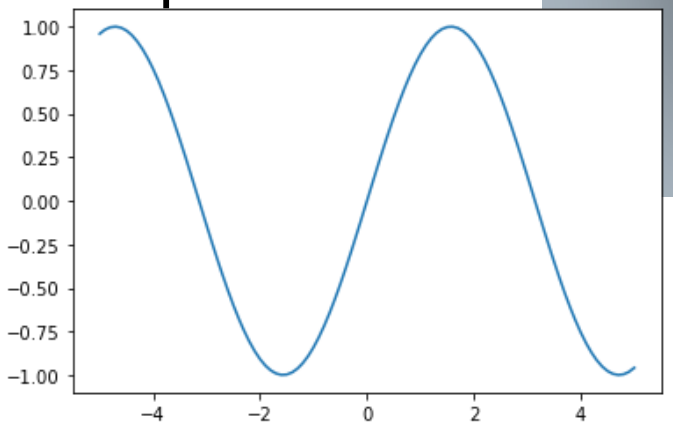
- We’re going to use visualization tool as a matplotlib mainly
- This library can print beautifully axis, label, deployment, color, etc…
- We’re going to use IDE as a Jupyter Notebook

Graph output using matplotlib

plt.plot(x1, y1)

- Import numpy to make data set
- Import matplotlib to make graph
- We set the range from -5 to 5 to x1 to output a graph of the sin function in a specific range.

```
%matplotlib inline

import numpy as np
import matplotlib.pyplot as plt

x1 = np.linspace(-5,5, 101)
y1 = np.sin(x1)

plt.plot(x1, y1)
```

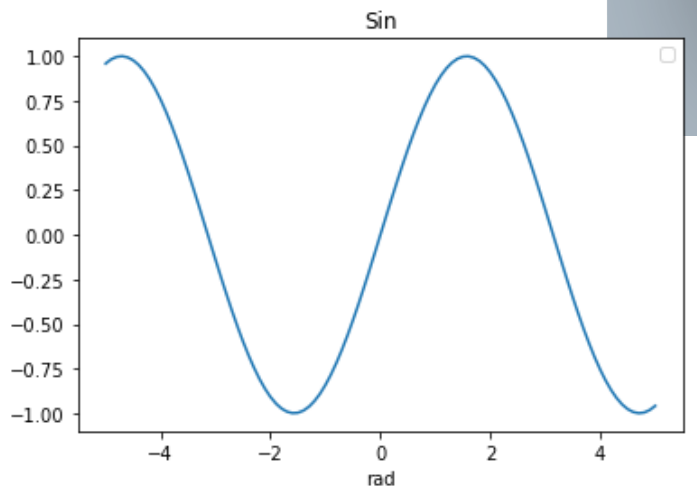


Graph output using matplotlib

ax.plot(x1, y1)

- fig is an object of the Figure class that generates a canvas, which is the entire area of the graph
- ax is an object of the Axes class that generates a coordinate plane.

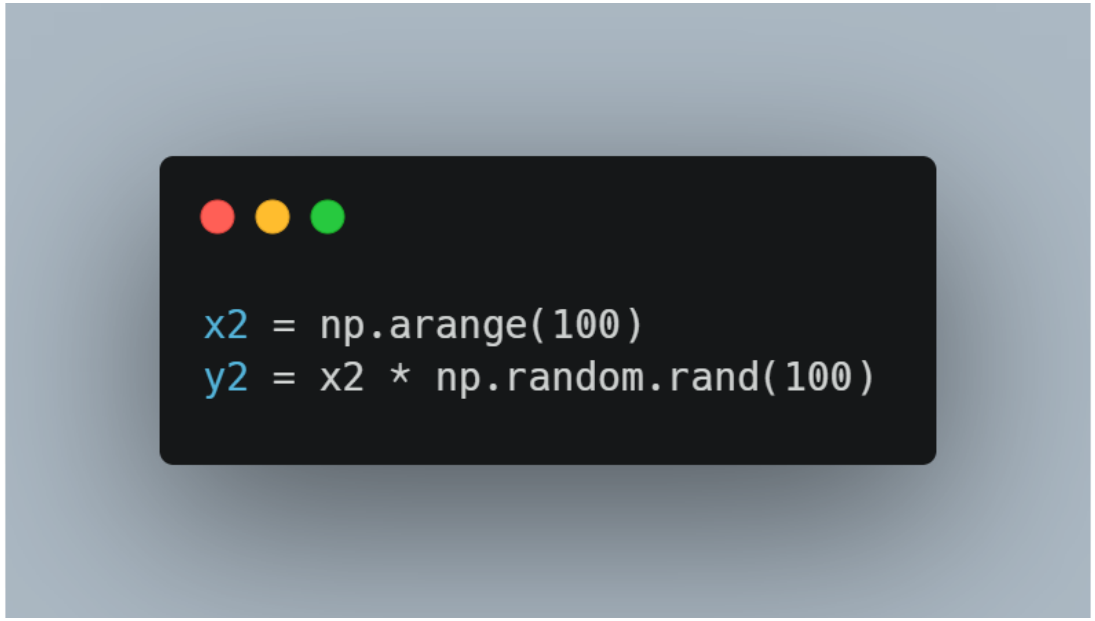
```
fig, ax = plt.subplots()

ax.set_title('Sin')
ax.set_xlabel('rad')
ax.plot(x1, y1)

plt.show()
```



Output various graphs

Preparing Data

- x2 stores an array from 0 to 99 as a number array function.
- y2 stores the result obtained by multiplying an array of 100 random numbers from 0 to 1 by x2.

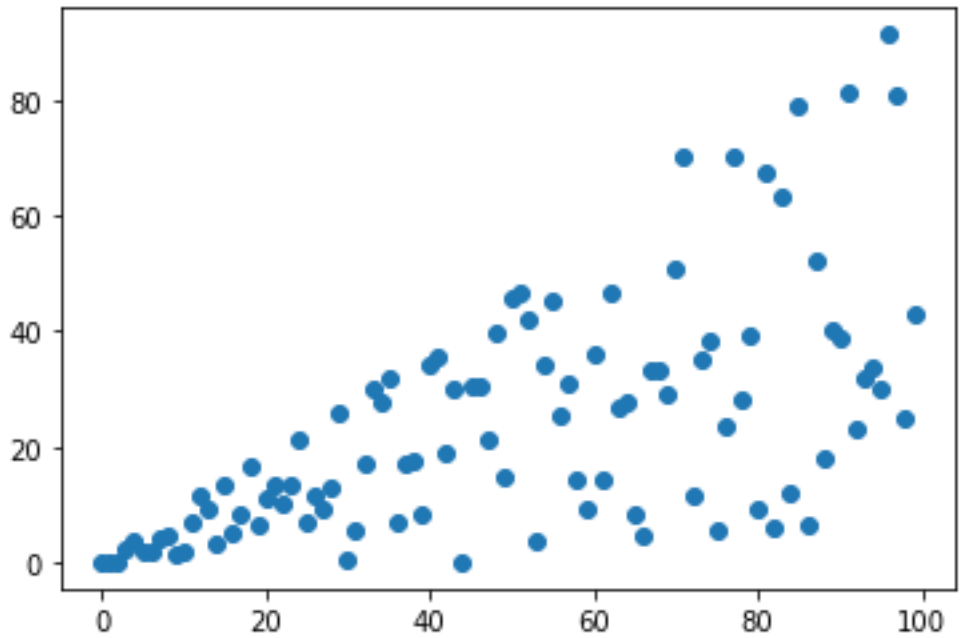


Select Graphs by Purpose

- Bar graph: The data you want to express must all have the same unit. Discontinuous variables are good.
- Line graph: Used for continuous data, such as changes over time
- Scatter: Used to express the relationship between irregular intervals or bundles of data
- Histogram: Use to determine the shape of the distribution
- Box plot: Use to easily understand the alternative distribution of data

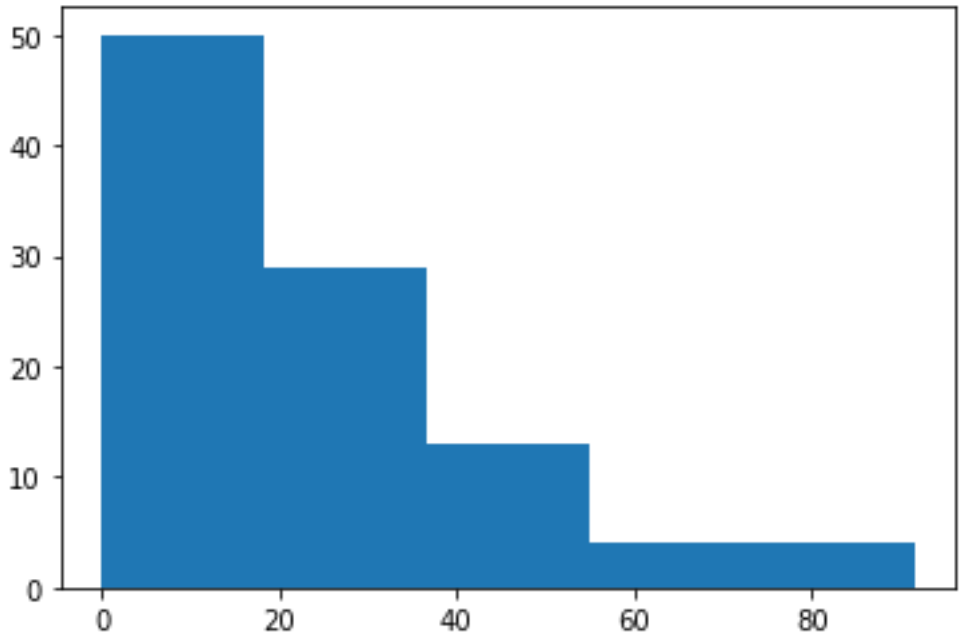
Output various graphs

scatter




```
plt.scatter(x2, y2)
```

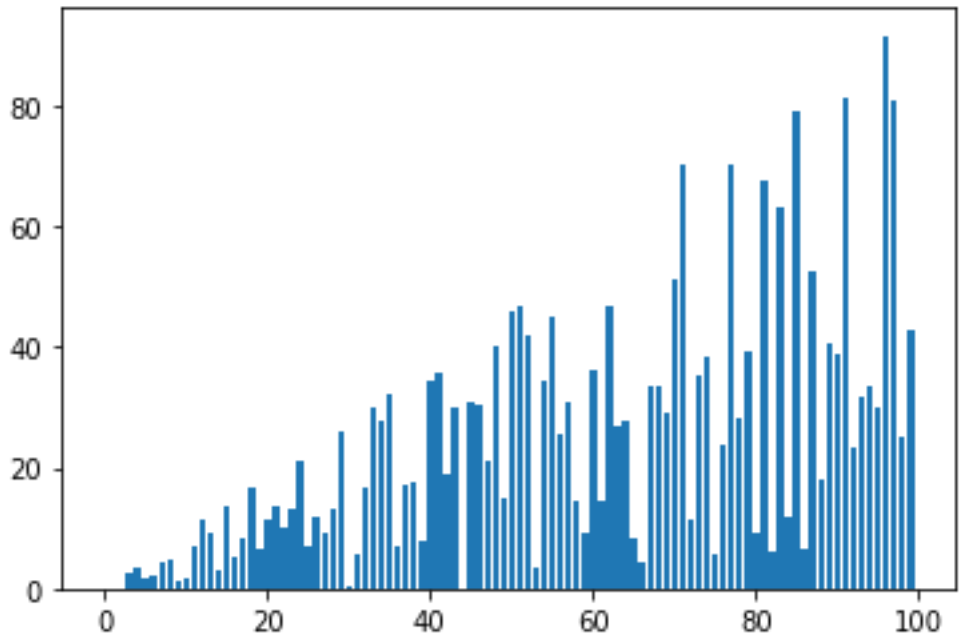
Output various graphs
histogram




```
plt.hist(y2, bins = 5)
```

Output various graphs

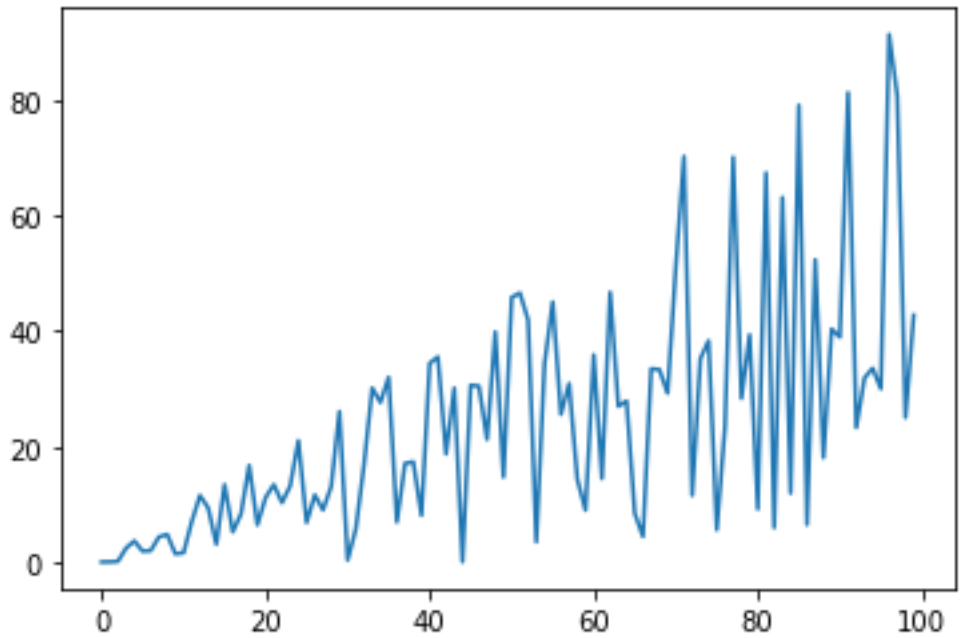
bar graph




```
plt.bar(x2, y2)
```

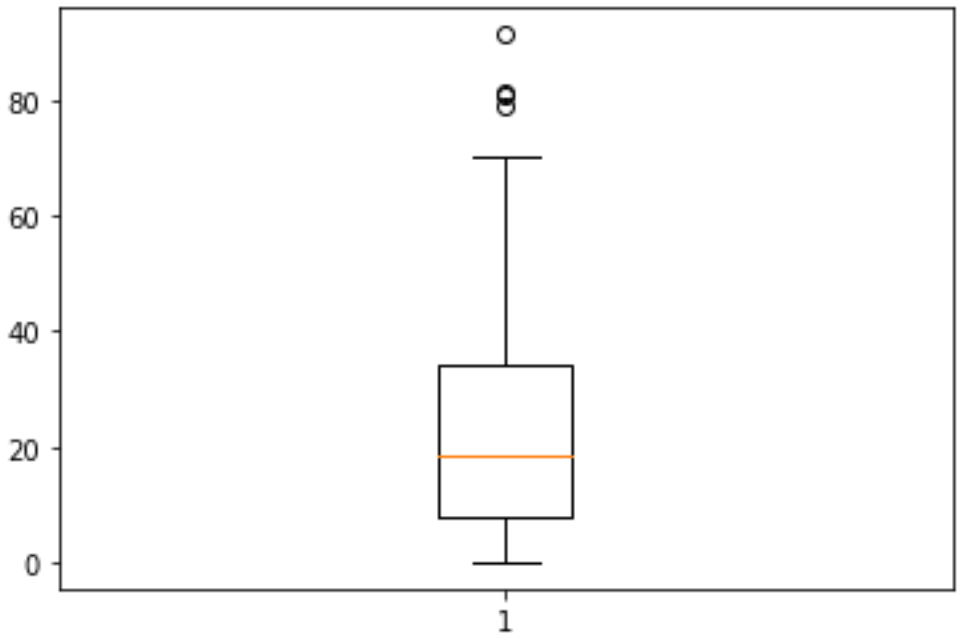
Output various graphs

line graph




```
plt.plot(x2, y2)
```

Output various graphs
box plot




```
plt.boxplot(y2)
```

Visualization of wine data set

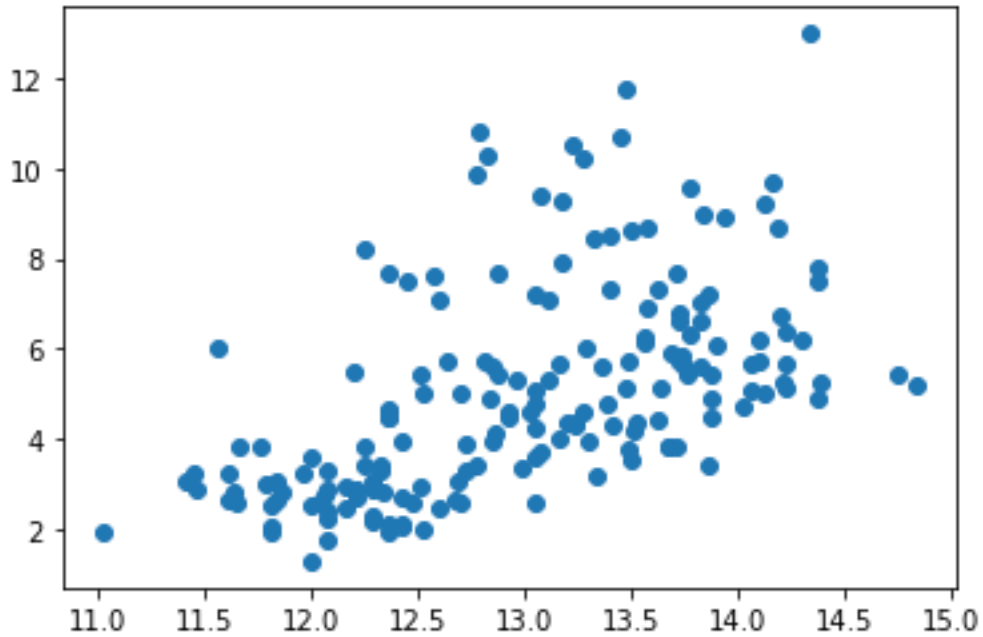
data preprocessing

- Import wine data set
- Get only the data we want

```
from sklearn.datasets import load_wine

data = load_wine()

x3 = data.data[:, [0]]
y3 = data.data[:, [9]]
```

Visualization of wine data set

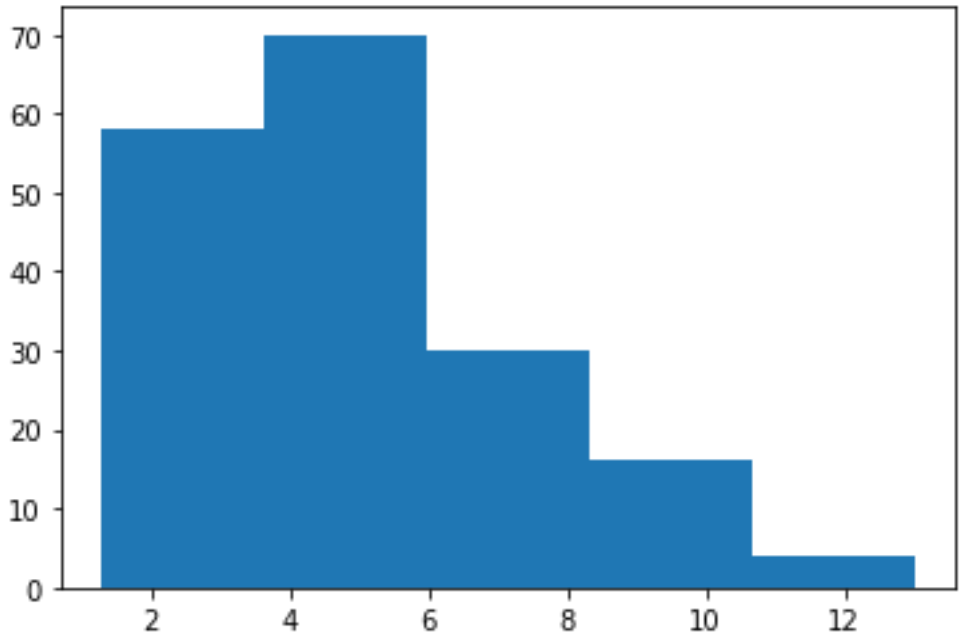
scatter




```
plt.scatter(x3, y3)
```

Visualization of wine data set

histogram




```
plt.hist(y3, bins = 5)
```

Use pandas to understand and handle data

- Import pandas
- Get wine datasets
- Make data frame using pandas
- Output a part of the dataset using the head method

```
import pandas as pd

from sklearn.datasets import load_wine

data = load_wine()
df_X = pd.DataFrame(data.data, columns = data.feature_names)

df_X.head()
```

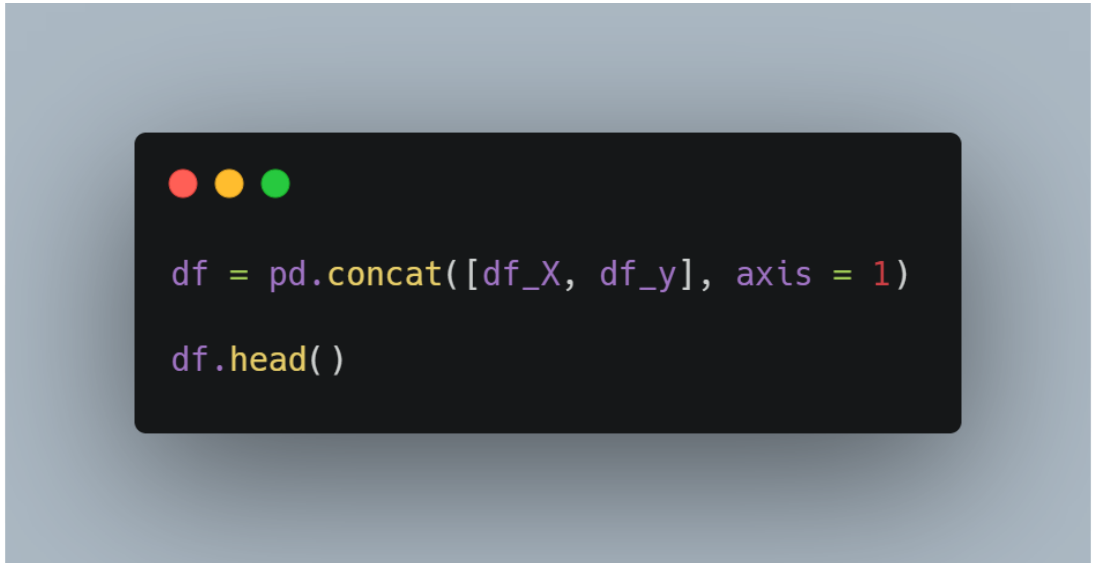
Use pandas to understand and handle data

- Set column label as a ‘kind(target)’
- Output a part of the dataset using the head method

```
df_y = pd.DataFrame(data.target, columns = ['kind(target)'])

df_y.head()
```

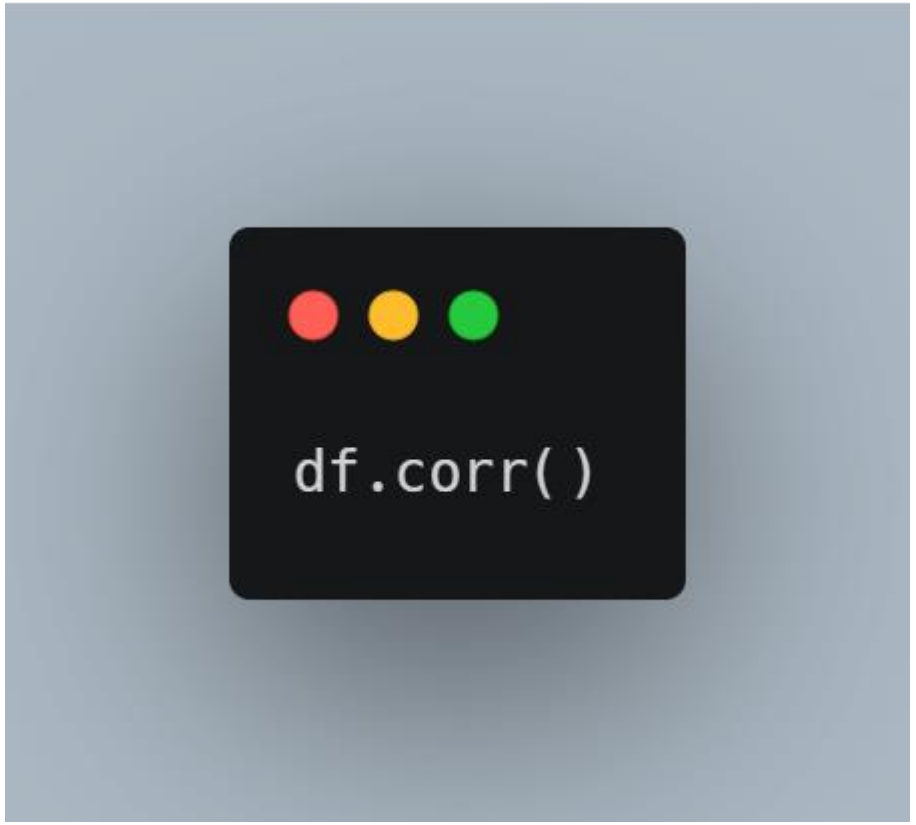
Use pandas to understand and handle data

- Combine target data and feature data
- Output a part of the dataset using the head method



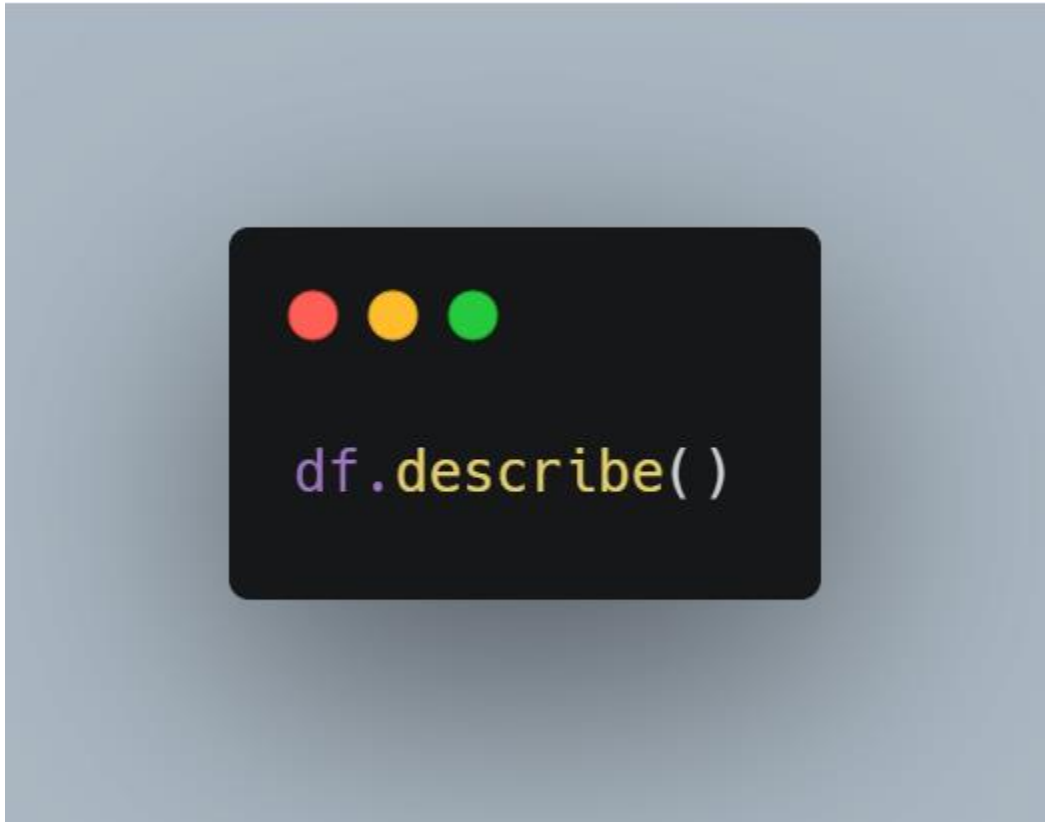
Use pandas to understand and handle data

- corr() function outputs correlation coefficient
- The correlation coefficient is a positive correlation as it approaches 1, and a negative correlation as it approaches -1, it is negative.
- If it is close to 0, there is no correlation between the data.

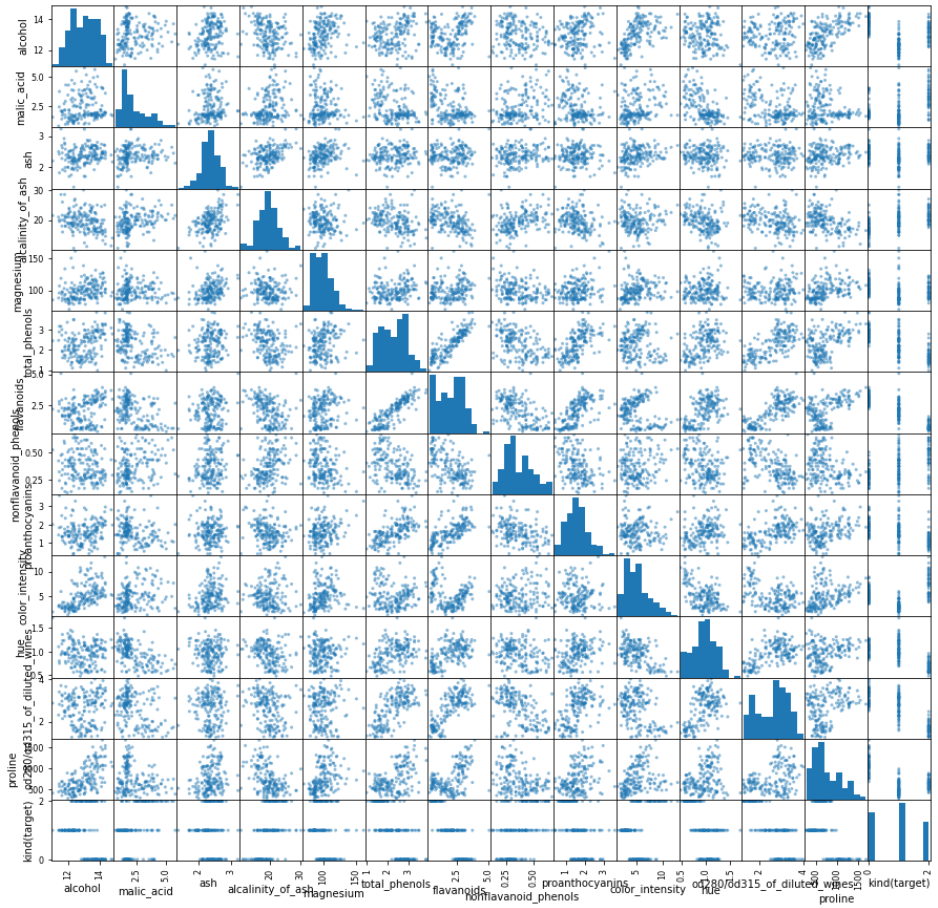


Use pandas to understand and handle data

- describe() function outputs eight pieces of statistical information.
- count, mean, std, min, 25%, 50%, 75%, max

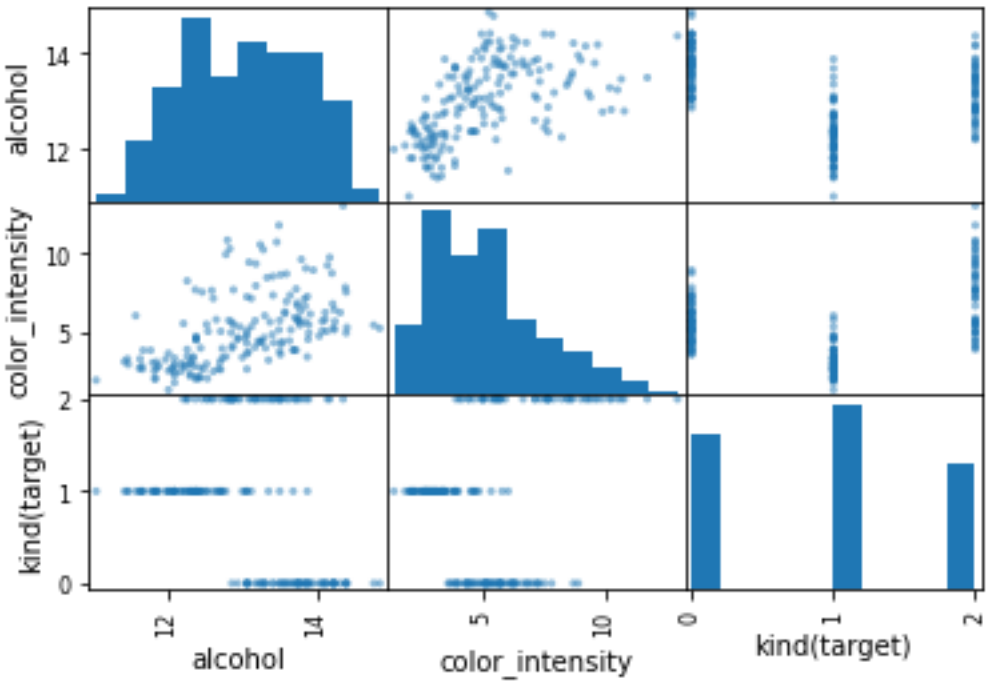


Use pandas to understand and handle data




```
from pandas.plotting import scatter_matrix

_ = scatter_matrix(df, figsize = (15, 15))
```

Use pandas to understand and handle data




```
_ = scatter_matrix(df.iloc[:, [0, 9, -1]])
```