COMP 4331 Tut 3

Jiaxin Xie

Outline

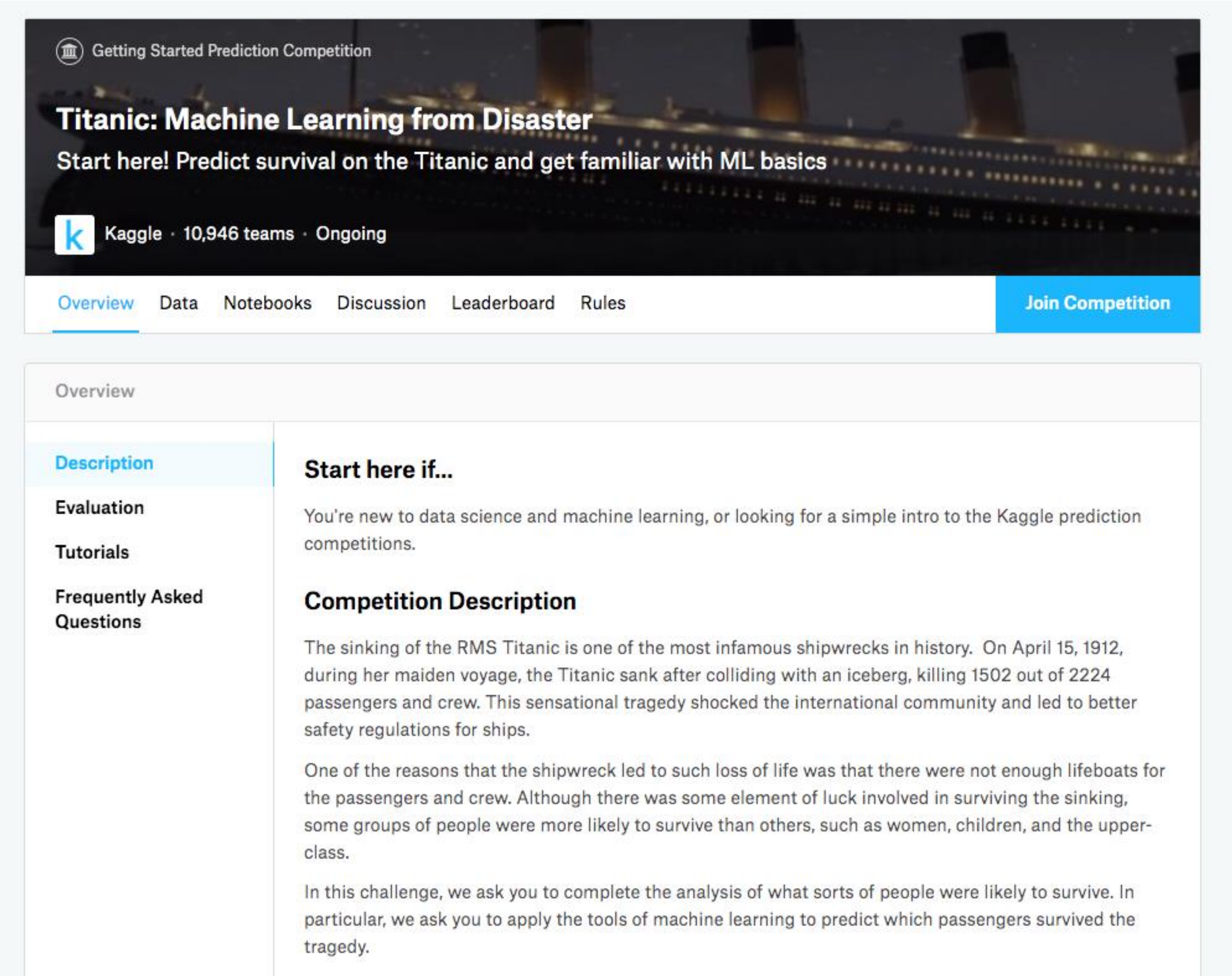
- Brief introduction to Kaggle
- Brief introduction to Pandas
- Demonstration of using pandas to preprocess the raw data
- Brief introduction to sklearn
- Review various plots

What is Kaggle

Kaggle is a competition platform, where you can find data, examples, tutorials, even courses.

- Free
- Some of the projects even provide computing environments
- You can ask people there
- For more information, you can refer to: https://www.kaggle.com

Titanic



Getting Started Prediction Competition

Titanic: Machine Learning from Disaster

Start here! Predict survival on the Titanic and get familiar with ML basics

Kaggle · 10,946 teams · Ongoing

Start here if...

You're new to data science and machine learning, or looking for a simple intro to the Kaggle prediction competitions.

Competition Description

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. On April 15, 1912, during her maiden voyage, the Titanic sank after colliding with an iceberg, killing 1502 out of 2224 passengers and crew. This sensational tragedy shocked the international community and led to better safety regulations for ships.

One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. Although there was some element of luck involved in surviving the sinking, some groups of people were more likely to survive than others, such as women, children, and the upper-class.

In this challenge, we ask you to complete the analysis of what sorts of people were likely to survive. In particular, we ask you to apply the tools of machine learning to predict which passengers survived the tragedy.

https://www.kaggle.com/c/titanic/overview

Experiment setup

- Start the Jupyter Notebook (jupyter notebook)
- Connect to it via your notebook via your browser.
- Create a new ipynb file (click ‘new’ button)
- In the first box, copy and paste all the codes in next slides to load used packages
- Press shift+enter to execute this box. (PS: you may get error saying that you cannot find a package, please install it in your server with “pip install [package name]”)

Experiment setup

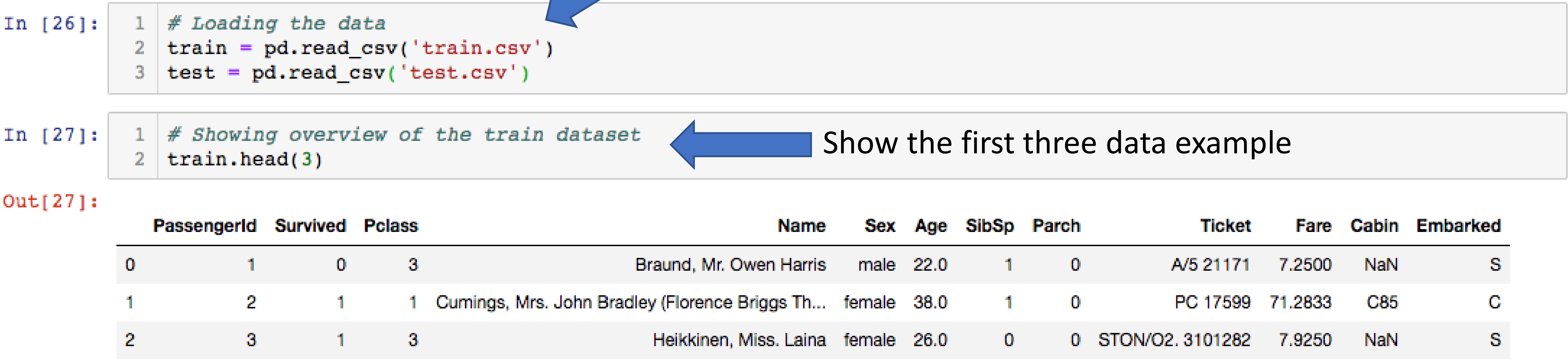
```
import numpy as np
import pandas as pd
from sklearn import tree
from sklearn.metrics import accuracy_score
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from IPython.display import Image as PImage
from subprocess import check_call
# import all the needed package
```

Introduction to Pandas

- Powerful and productive python data analysis and management library.
- Short name of **Pan**el **Da**ta **S**ystem
- Open sourced by AQR capital management, LLC in late 2009
- Used by both academic and industry.

Load data with pandas

Load the train and test data



In [26]:

```
# Loading the data
train = pd.read_csv('train.csv')
test = pd.read_csv('test.csv')
```

In [27]:

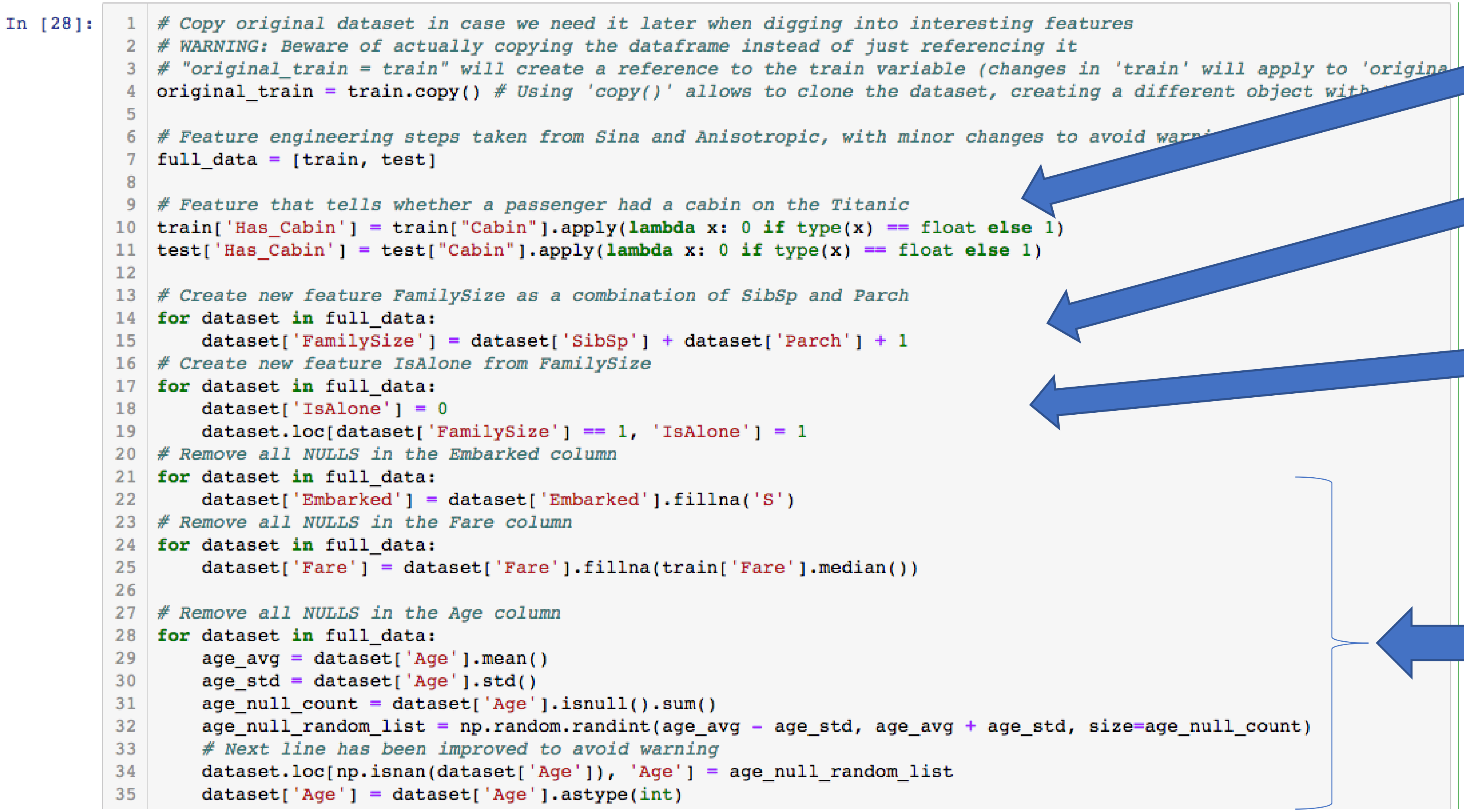
```
# Showing overview of the train dataset
train.head(3)
```

Show the first three data example

Out[27]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S

Preprocess data with Pandas (1)



```
In [28]:
# Copy original dataset in case we need it later when digging into interesting features
# WARNING: Beware of actually copying the dataframe instead of just referencing it
# "original_train = train" will create a reference to the train variable (changes in 'train' will apply to 'origina
original_train = train.copy() # Using 'copy()' allows to clone the dataset, creating a different object with

# Feature engineering steps taken from Sina and Anisotropic, with minor changes to avoid warn
full_data = [train, test]

# Feature that tells whether a passenger had a cabin on the Titanic
train['Has_Cabin'] = train["Cabin"].apply(lambda x: 0 if type(x) == float else 1)
test['Has_Cabin'] = test["Cabin"].apply(lambda x: 0 if type(x) == float else 1)

# Create new feature FamilySize as a combination of SibSp and Parch
for dataset in full_data:
    dataset['FamilySize'] = dataset['SibSp'] + dataset['Parch'] + 1
# Create new feature IsAlone from FamilySize
for dataset in full_data:
    dataset['IsAlone'] = 0
    dataset.loc[dataset['FamilySize'] == 1, 'IsAlone'] = 1
# Remove all NULLS in the Embarked column
for dataset in full_data:
    dataset['Embarked'] = dataset['Embarked'].fillna('S')
# Remove all NULLS in the Fare column
for dataset in full_data:
    dataset['Fare'] = dataset['Fare'].fillna(train['Fare'].median())

# Remove all NULLS in the Age column
for dataset in full_data:
    age_avg = dataset['Age'].mean()
    age_std = dataset['Age'].std()
    age_null_count = dataset['Age'].isnull().sum()
    age_null_random_list = np.random.randint(age_avg - age_std, age_avg + age_std, size=age_null_count)
    # Next line has been improved to avoid warning
    dataset.loc[np.isnan(dataset['Age']), 'Age'] = age_null_random_list
    dataset['Age'] = dataset['Age'].astype(int)
```

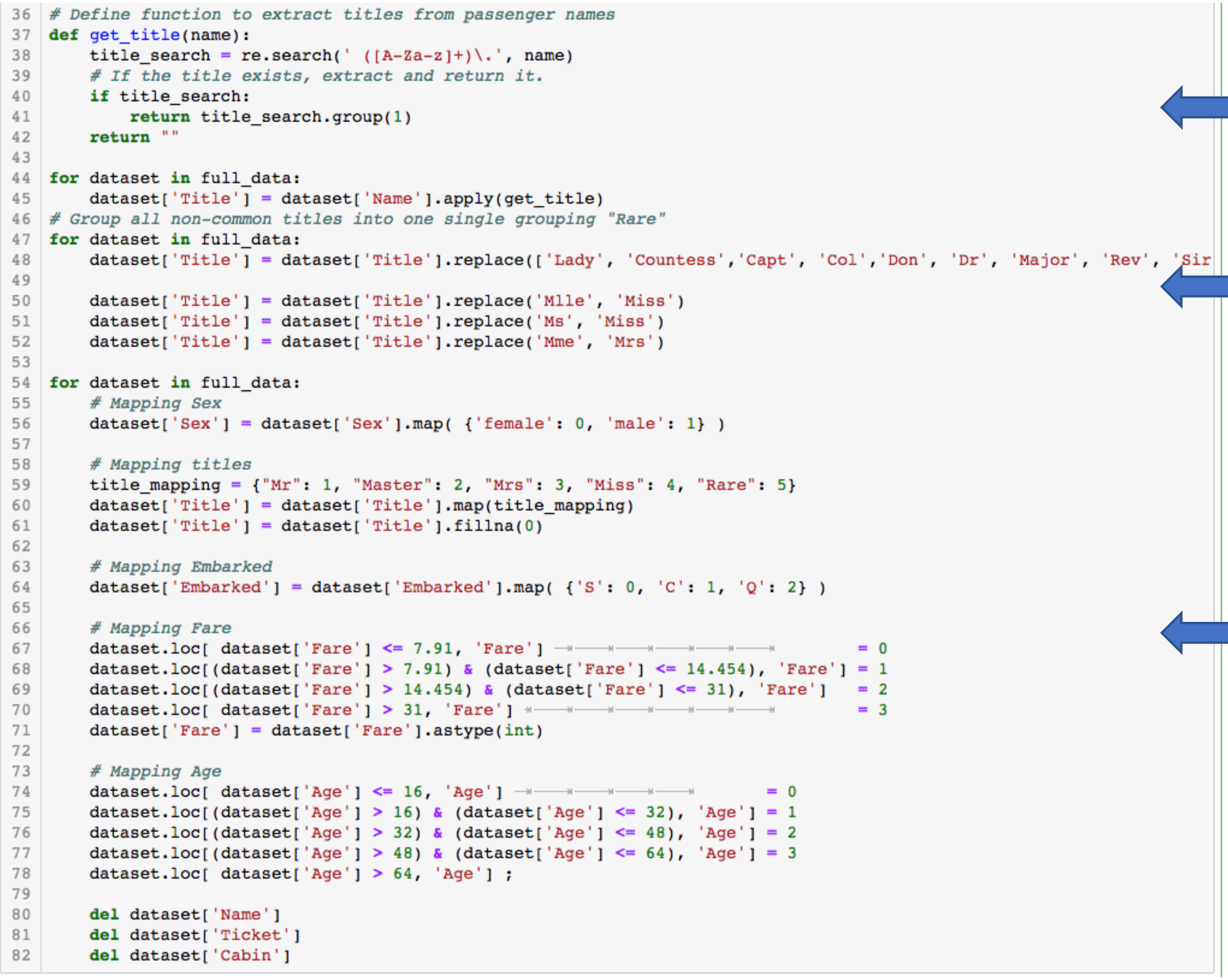
Put the train and test data together

Convert “family size” feature to integer

Create a new feature “IsAlone”

Replace empty data with average

Preprocess data with Pandas (2)



```python
# Define function to extract titles from passenger names
def get_title(name):
    title_search = re.search(' ([A-Za-z]+)\.', name)
    # If the title exists, extract and return it.
    if title_search:
        return title_search.group(1)
    return ""

for dataset in full_data:
    dataset['Title'] = dataset['Name'].apply(get_title)
# Group all non-common titles into one single grouping "Rare"
for dataset in full_data:
    dataset['Title'] = dataset['Title'].replace(['Lady', 'Countess','Capt', 'Col','Don', 'Dr', 'Major', 'Rev', 'Sir

    dataset['Title'] = dataset['Title'].replace('Mlle', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Ms', 'Miss')
    dataset['Title'] = dataset['Title'].replace('Mme', 'Mrs')

for dataset in full_data:
    # Mapping Sex
    dataset['Sex'] = dataset['Sex'].map( {'female': 0, 'male': 1} )

    # Mapping titles
    title_mapping = {"Mr": 1, "Master": 2, "Mrs": 3, "Miss": 4, "Rare": 5}
    dataset['Title'] = dataset['Title'].map(title_mapping)
    dataset['Title'] = dataset['Title'].fillna(0)

    # Mapping Embarked
    dataset['Embarked'] = dataset['Embarked'].map( {'S': 0, 'C': 1, 'Q': 2} )

    # Mapping Fare
    dataset.loc[ dataset['Fare'] <= 7.91, 'Fare']                                   = 0
    dataset.loc[(dataset['Fare'] > 7.91) & (dataset['Fare'] <= 14.454), 'Fare'] = 1
    dataset.loc[(dataset['Fare'] > 14.454) & (dataset['Fare'] <= 31), 'Fare']   = 2
    dataset.loc[ dataset['Fare'] > 31, 'Fare']                                      = 3
    dataset['Fare'] = dataset['Fare'].astype(int)

    # Mapping Age
    dataset.loc[ dataset['Age'] <= 16, 'Age']                          = 0
    dataset.loc[(dataset['Age'] > 16) & (dataset['Age'] <= 32), 'Age'] = 1
    dataset.loc[(dataset['Age'] > 32) & (dataset['Age'] <= 48), 'Age'] = 2
    dataset.loc[(dataset['Age'] > 48) & (dataset['Age'] <= 64), 'Age'] = 3
    dataset.loc[ dataset['Age'] > 64, 'Age'] ;

    del dataset['Name']
    del dataset['Ticket']
    del dataset['Cabin']
```

Define a function to clean the data

Clean all the title feature

Replace all the feature with number

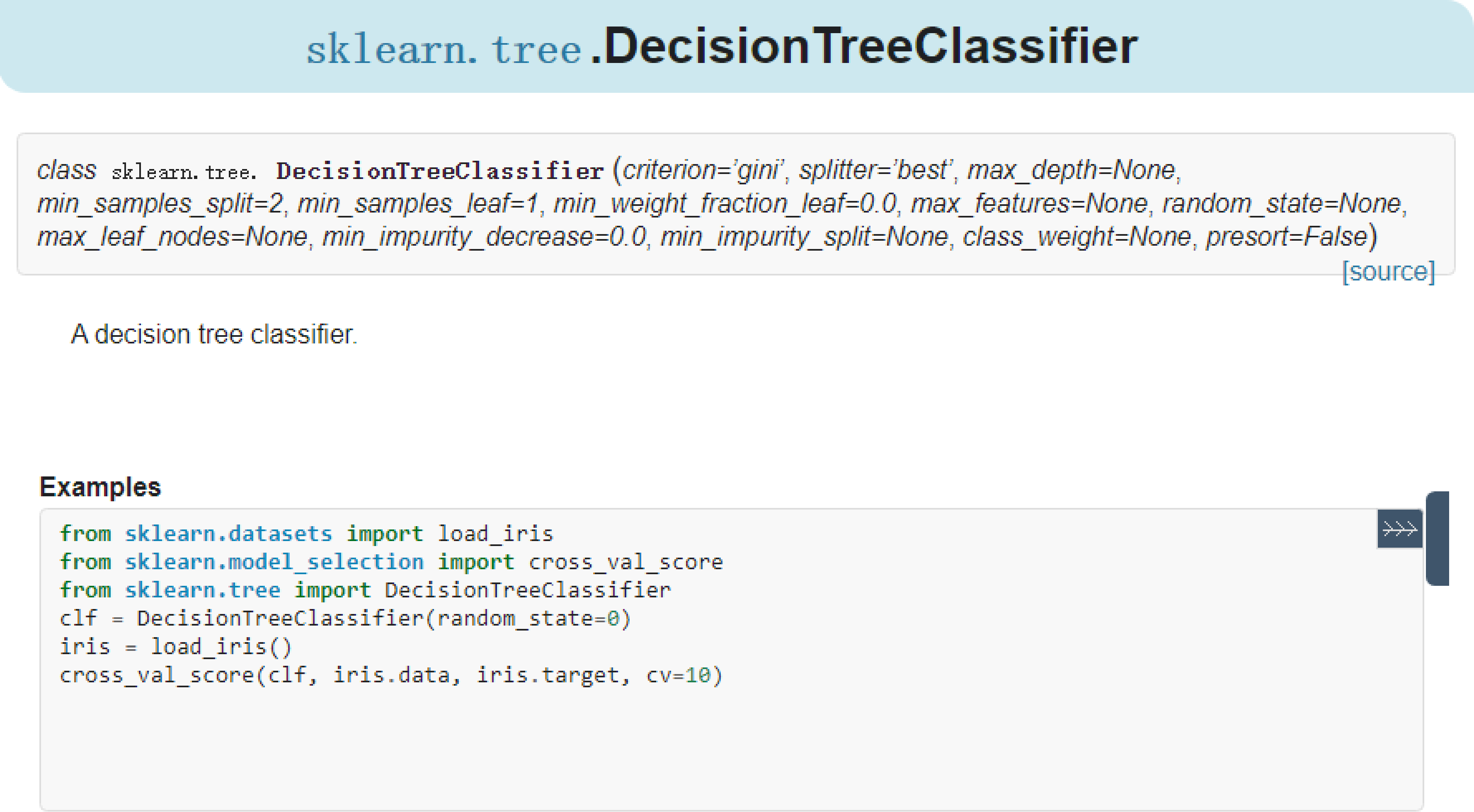
Introduction to scikit-learn (sklearn)

Extensions to Scipy (Scientific Python) are called Scikits. Scikie-learn provides machine learning algorithm.

- Algorithms for supervised and unsupervised learning
- Built on Scipy and Numpy
- Standard Python API interface
- Sits on top of c libraries, LAPACK, LibSVM, and Cython
- Open Source

For more information, you can refer to: https://scikit-learn.org/stable/

Build a decision tree model with sklearn



sklearn.tree.DecisionTreeClassifier

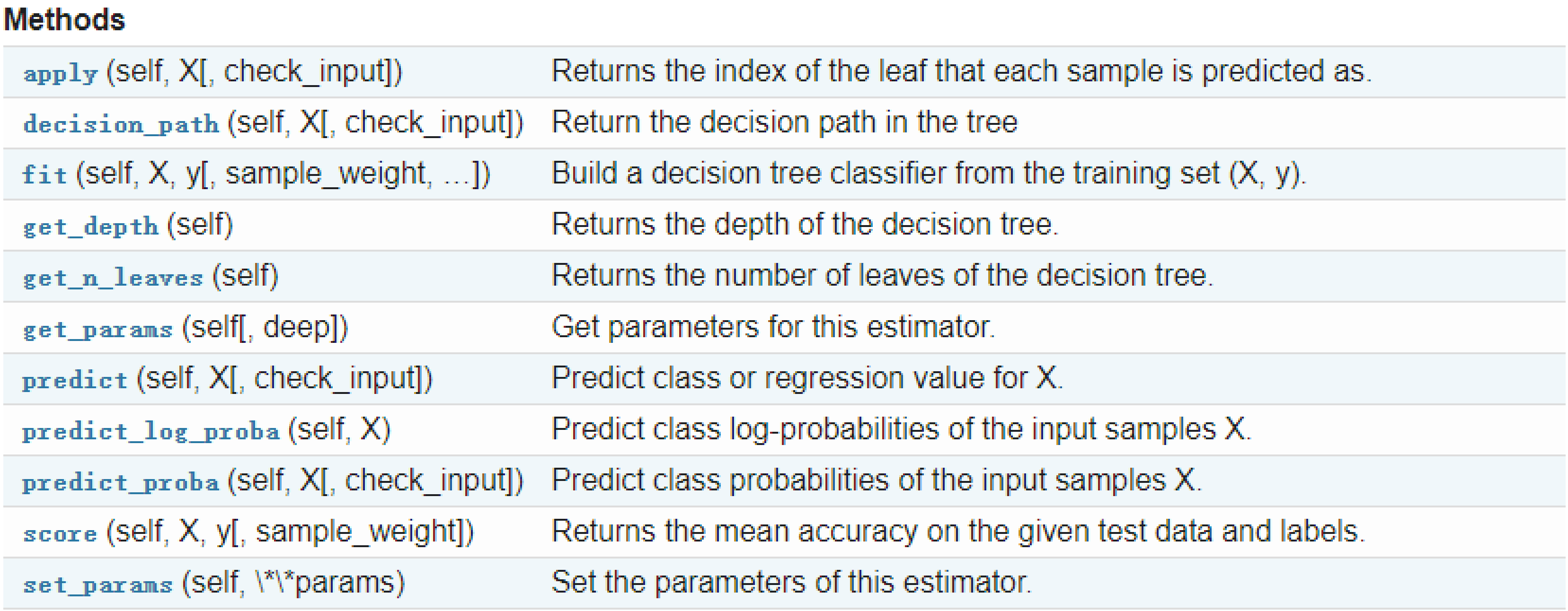
class sklearn.tree. **DecisionTreeClassifier** (*criterion='gini', splitter='best', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0.0, min_impurity_split=None, class_weight=None, presort=False*) [source]

A decision tree classifier.

Examples

```
from sklearn.datasets import load_iris
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
clf = DecisionTreeClassifier(random_state=0)
iris = load_iris()
cross_val_score(clf, iris.data, iris.target, cv=10)
```

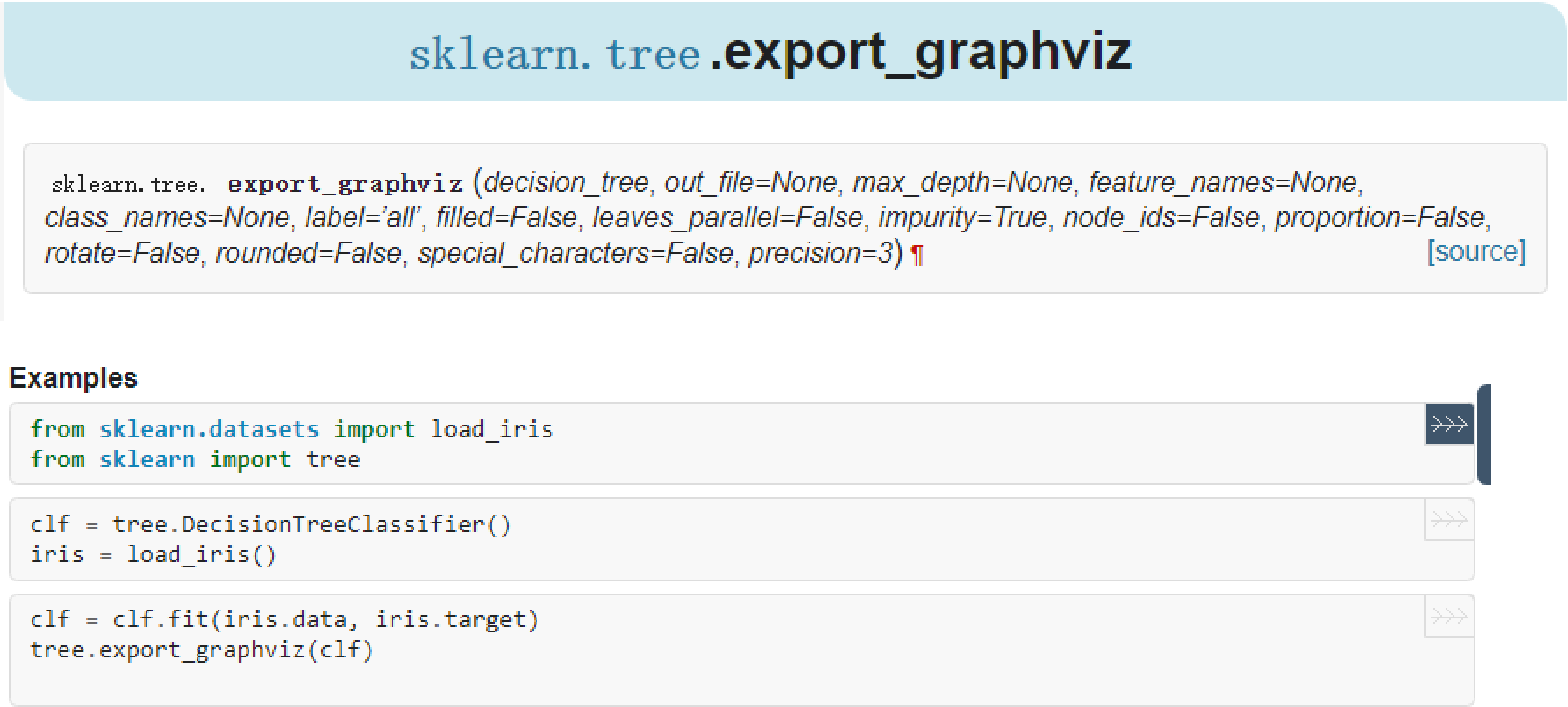
Build a decision tree model with sklearn



Methods

apply (self, X[, check_input])	Returns the index of the leaf that each sample is predicted as.
decision_path (self, X[, check_input])	Return the decision path in the tree
fit (self, X, y[, sample_weight, …])	Build a decision tree classifier from the training set (X, y).
get_depth (self)	Returns the depth of the decision tree.
get_n_leaves (self)	Returns the number of leaves of the decision tree.
get_params (self[, deep])	Get parameters for this estimator.
predict (self, X[, check_input])	Predict class or regression value for X.
predict_log_proba (self, X)	Predict class log-probabilities of the input samples X.
predict_proba (self, X[, check_input])	Predict class probabilities of the input samples X.
score (self, X, y[, sample_weight])	Returns the mean accuracy on the given test data and labels.
set_params (self, **params)	Set the parameters of this estimator.

Visualize Decision Tree



sklearn.tree.export_graphviz

sklearn.tree. **export_graphviz** (*decision_tree*, *out_file=None*, *max_depth=None*, *feature_names=None*, *class_names=None*, *label='all'*, *filled=False*, *leaves_parallel=False*, *impurity=True*, *node_ids=False*, *proportion=False*, *rotate=False*, *rounded=False*, *special_characters=False*, *precision=3*) ¶ [source]

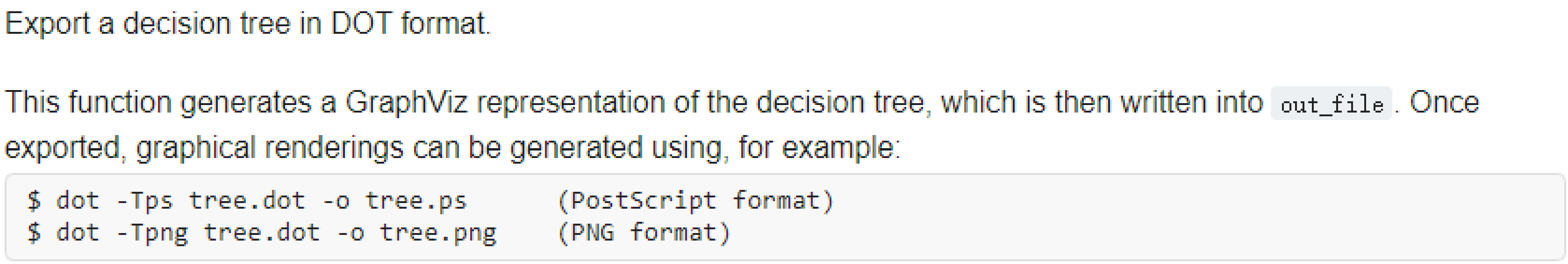
Examples

```
from sklearn.datasets import load_iris
from sklearn import tree
```

```
clf = tree.DecisionTreeClassifier()
iris = load_iris()
```

```
clf = clf.fit(iris.data, iris.target)
tree.export_graphviz(clf)
```

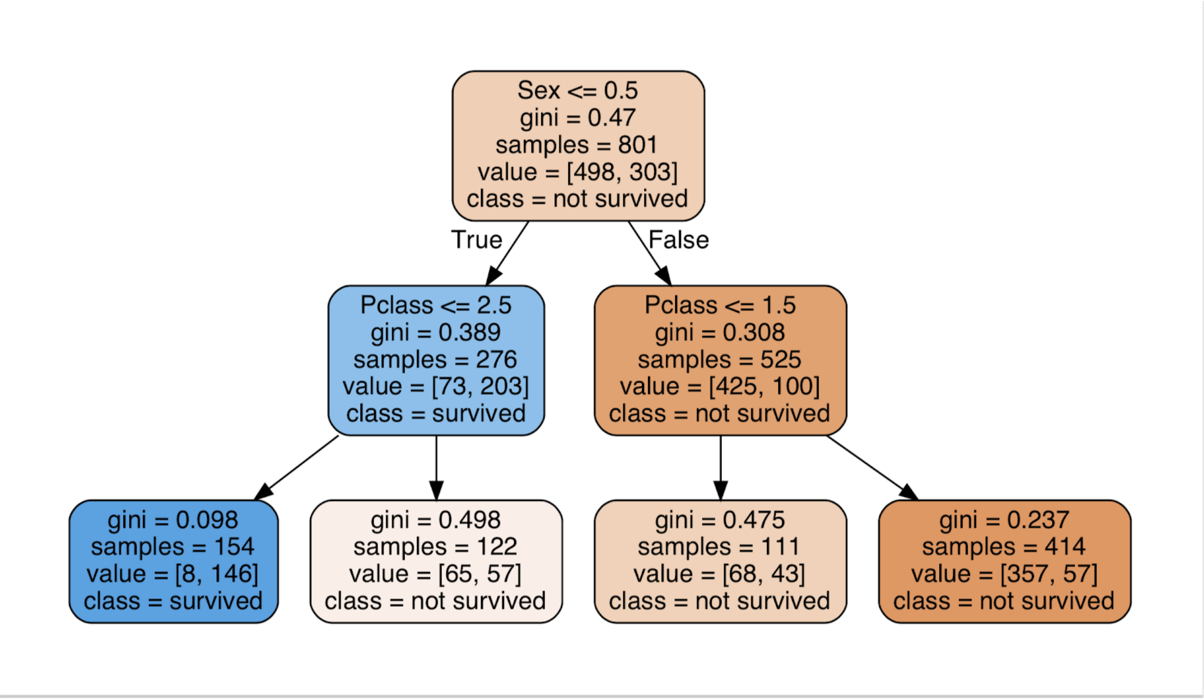
Visualize Decision Tree



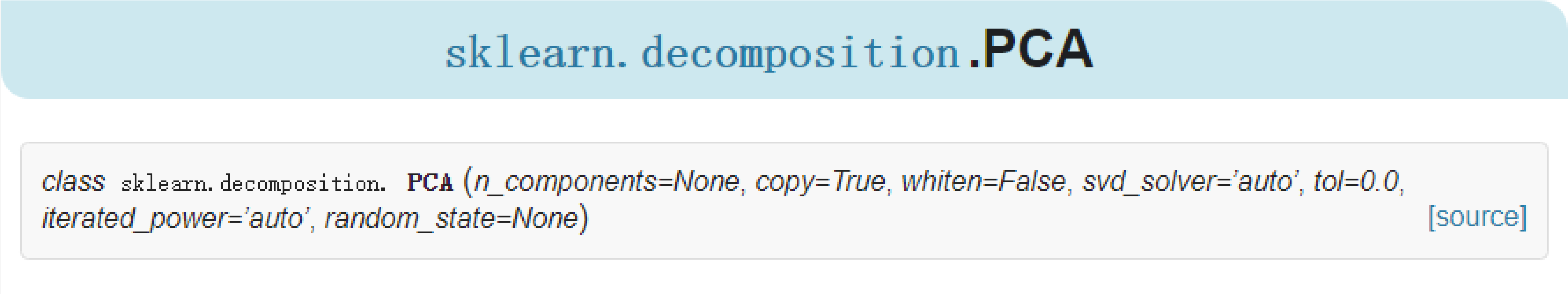
Export a decision tree in DOT format.

This function generates a GraphViz representation of the decision tree, which is then written into `out_file`. Once exported, graphical renderings can be generated using, for example:

```
$ dot -Tps tree.dot -o tree.ps      (PostScript format)
$ dot -Tpng tree.dot -o tree.png    (PNG format)
```

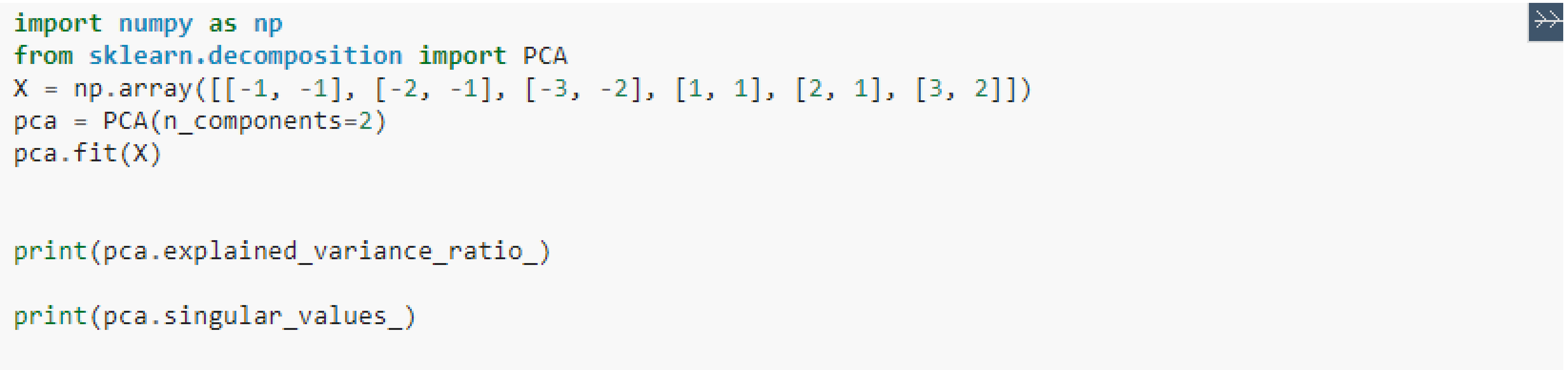


Implement PCA



sklearn.decomposition.PCA

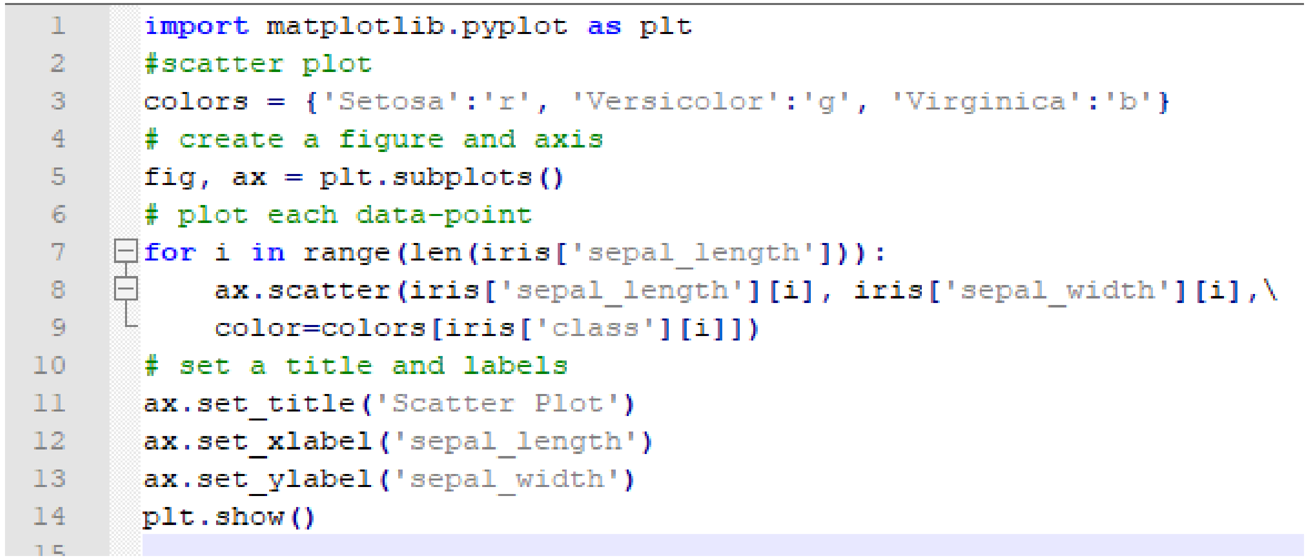
class sklearn.decomposition. **PCA** (*n_components=None, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None*) [source]



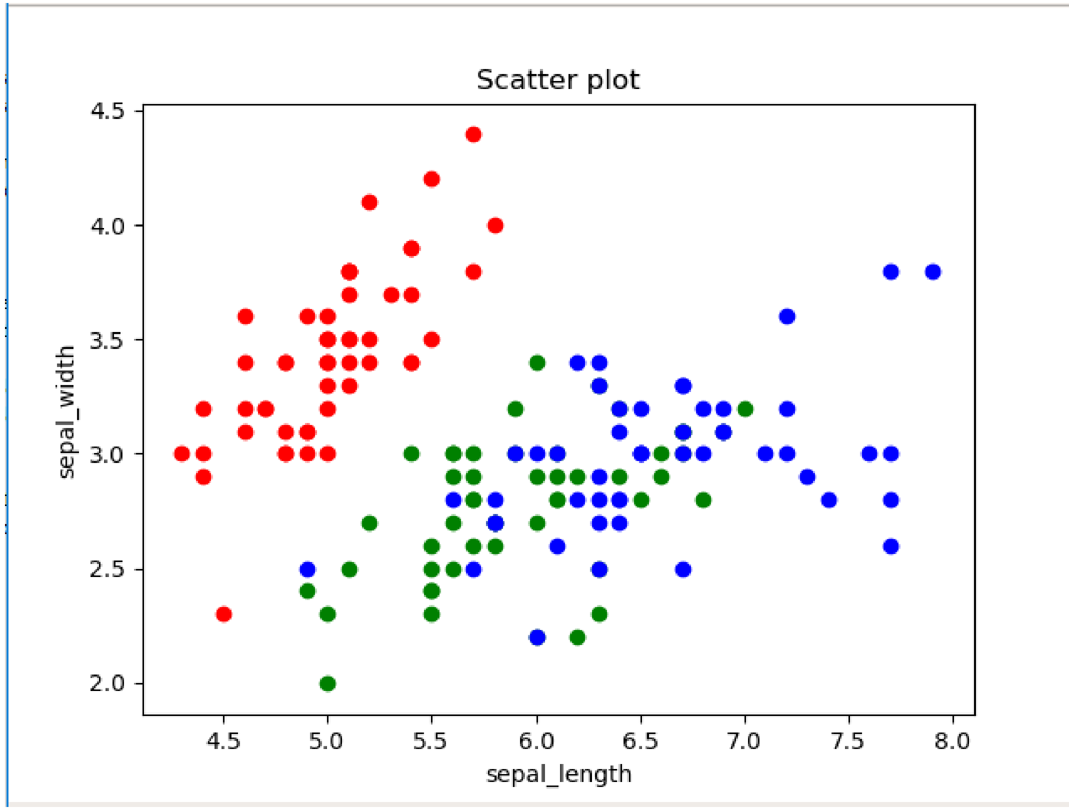
```
import numpy as np
from sklearn.decomposition import PCA
X = np.array([[-1, -1], [-2, -1], [-3, -2], [1, 1], [2, 1], [3, 2]])
pca = PCA(n_components=2)
pca.fit(X)


print(pca.explained_variance_ratio_)

print(pca.singular_values_)
```

In tutorial 1, we also release code to implement PCA.

Scatter plot

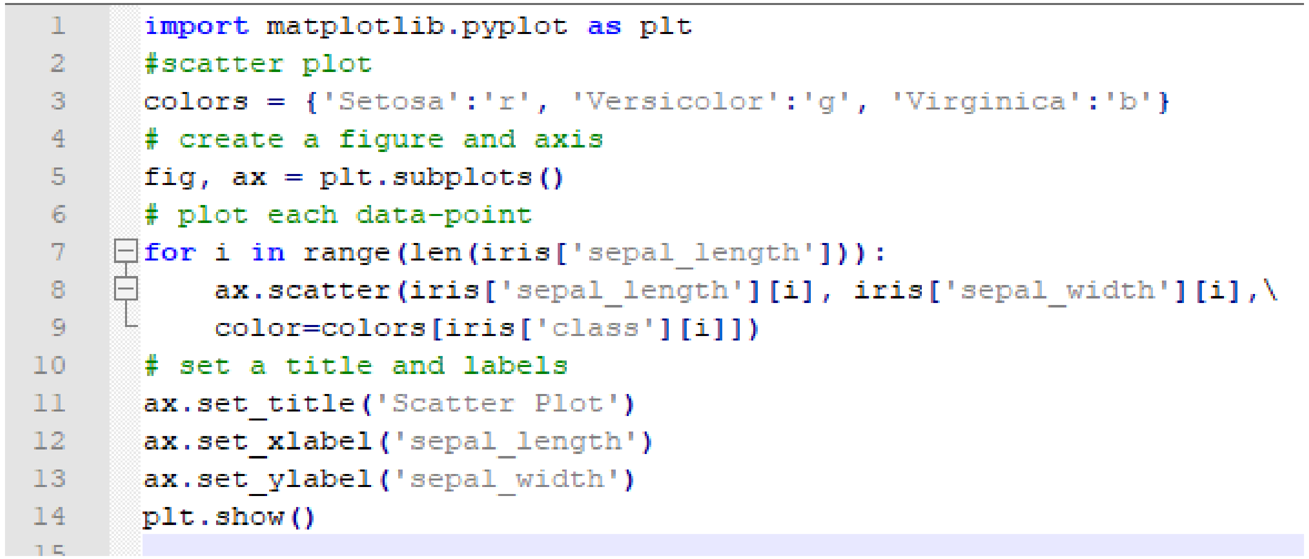


```python
import matplotlib.pyplot as plt
#scatter plot
colors = {'Setosa':'r', 'Versicolor':'g', 'Virginica':'b'}
# create a figure and axis
fig, ax = plt.subplots()
# plot each data-point
for i in range(len(iris['sepal_length'])):
    ax.scatter(iris['sepal_length'][i], iris['sepal_width'][i],\
    color=colors[iris['class'][i]])
# set a title and labels
ax.set_title('Scatter Plot')
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
plt.show()
```

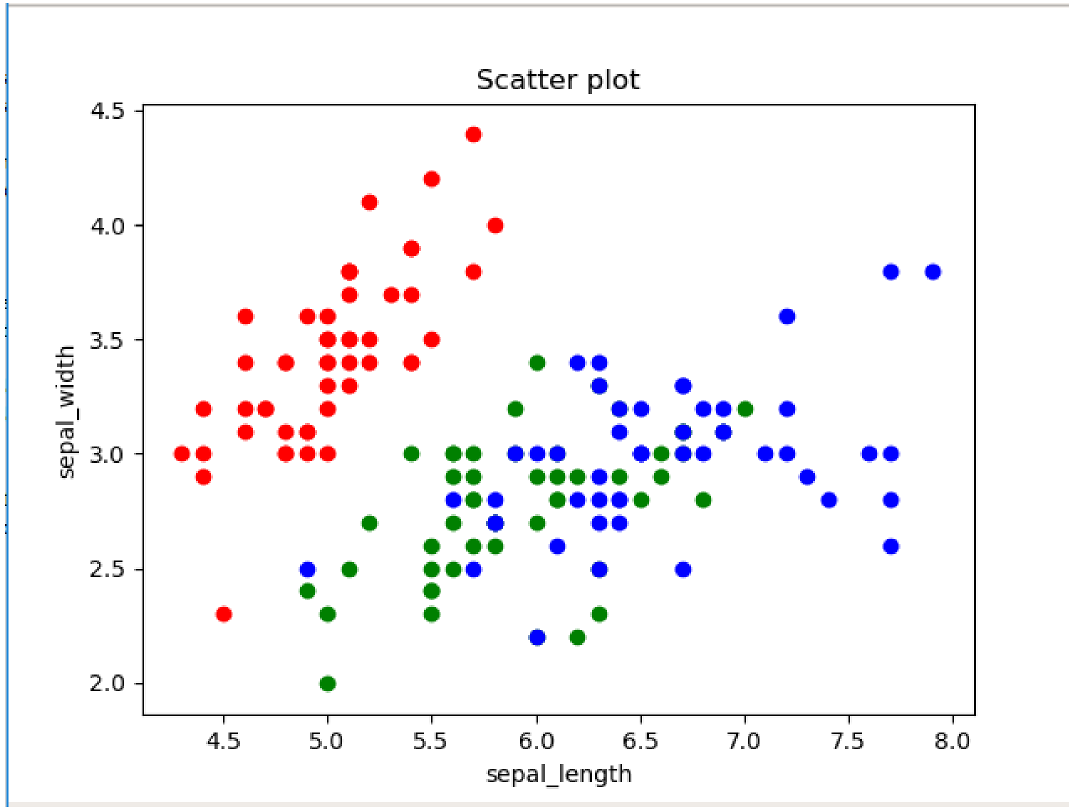


How to draw a scatter plot with sepal_length for the x-axis and petal_length for the y_axis?

Scatter plot

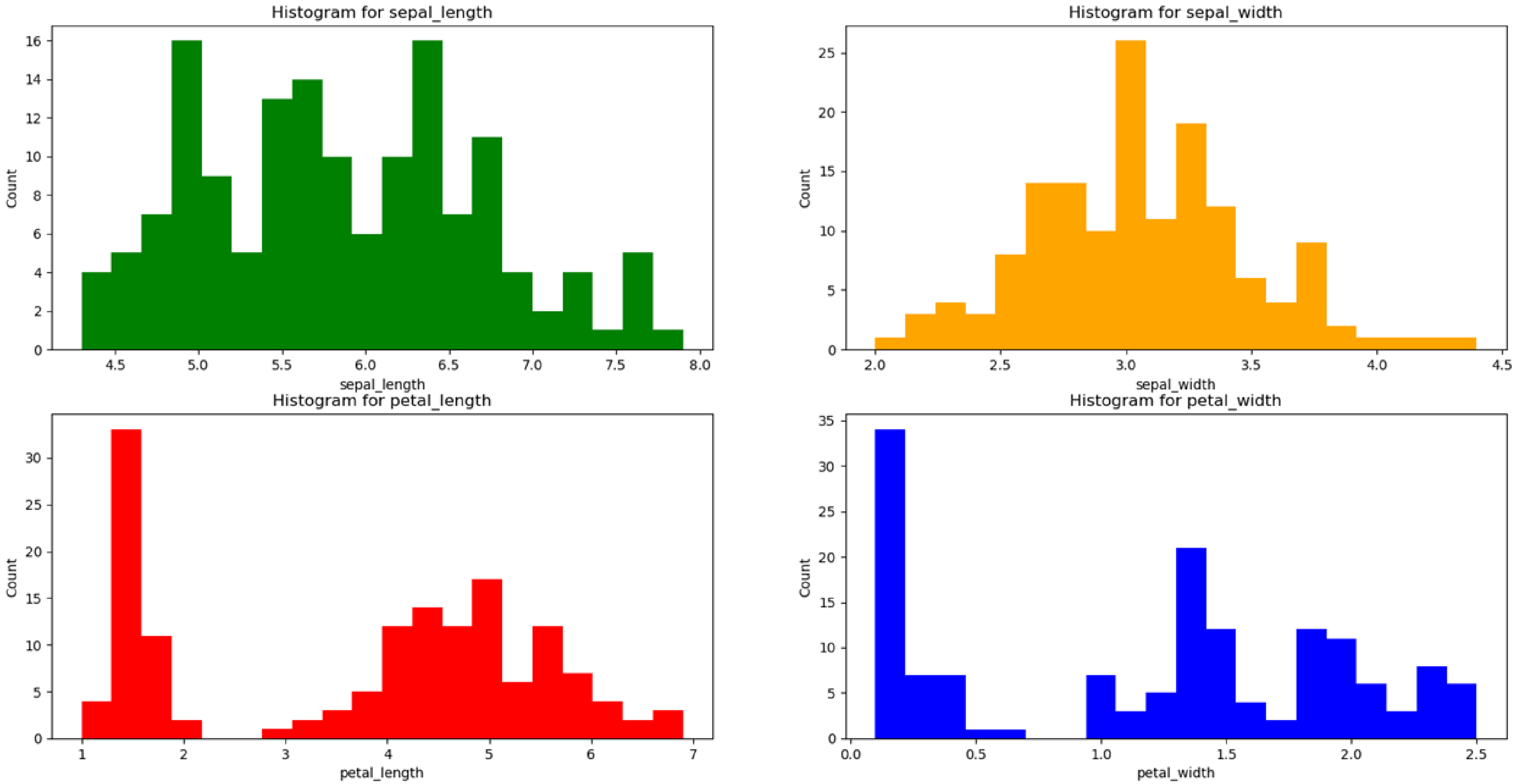


```
import matplotlib.pyplot as plt
#scatter plot
colors = {'Setosa':'r', 'Versicolor':'g', 'Virginica':'b'}
# create a figure and axis
fig, ax = plt.subplots()
# plot each data-point
for i in range(len(iris['sepal_length'])):
    ax.scatter(iris['sepal_length'][i], iris['sepal_width'][i],\
    color=colors[iris['class'][i]])
# set a title and labels
ax.set_title('Scatter Plot')
ax.set_xlabel('sepal_length')
ax.set_ylabel('sepal_width')
plt.show()
```

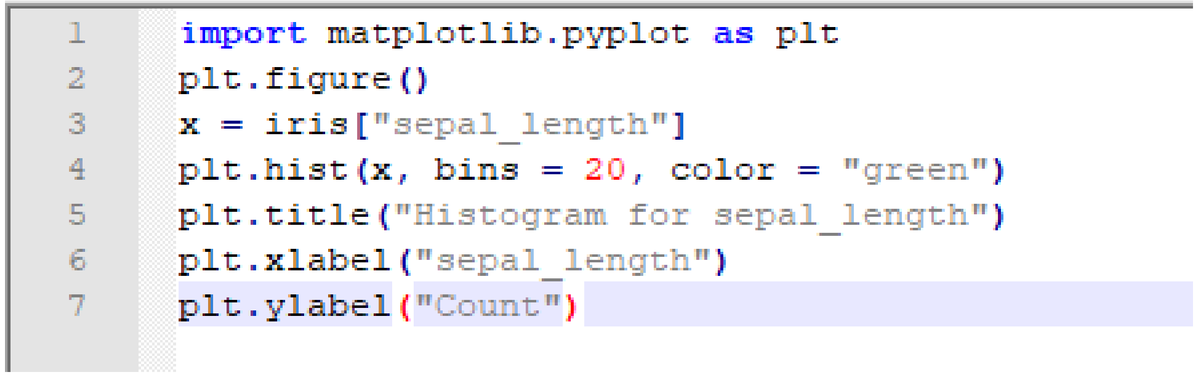


How to draw a scatter plot with sepal_length for the x-axis and petal_length for the y_axis?

Histogram


Histogram for sepal_length
Count
0
2
4
6
8
10
12
14
16
4.5
5.0
5.5
6.0
6.5
7.0
7.5
8.0
sepal_length
Histogram for sepal_width
Count
0
5
10
15
20
25
2.0
2.5
3.0
3.5
4.0
4.5
sepal_width
Histogram for petal_length
Count
0
5
10
15
20
25
30
1
2
3
4
5
6
7
petal_length
Histogram for petal_width
Count
0
5
10
15
20
25
30
35
0.0
0.5
1.0
1.5
2.0
2.5
petal_width

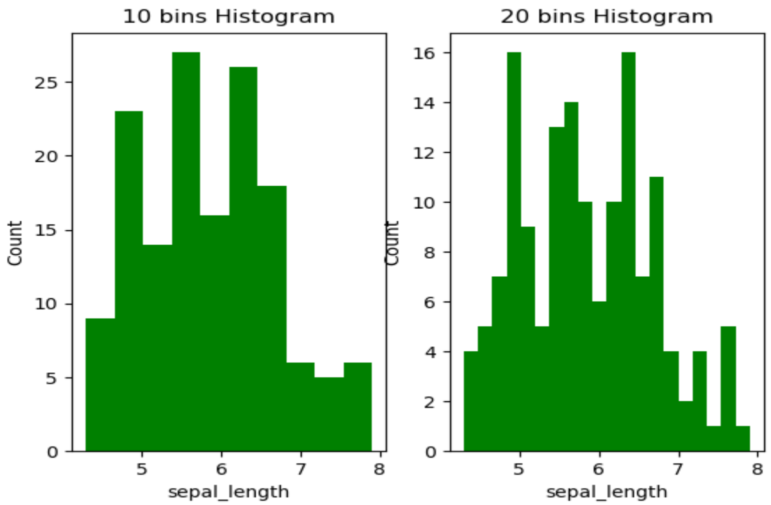
Histogram



```python
import matplotlib.pyplot as plt
plt.figure()
x = iris["sepal_length"]
plt.hist(x, bins = 20, color = "green")
plt.title("Histogram for sepal_length")
plt.xlabel("sepal_length")
plt.ylabel("Count")
```

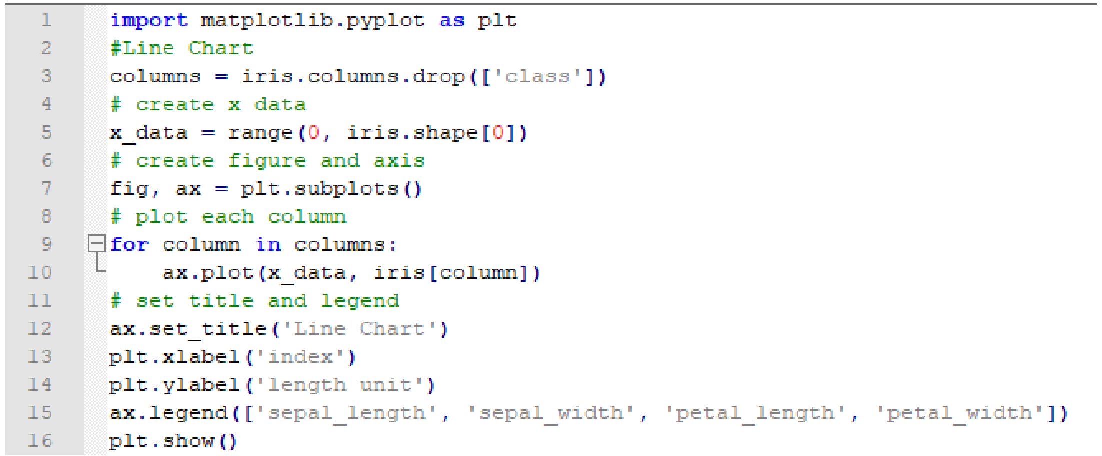
Function: matplotlib.pyplot.hist More details plz see https://matplotlib.org/3.1.1/api/_as_gen/matplotlib.pyplot.hist.html

Change bins? plt.hist(x, bins = 10, color = "green")

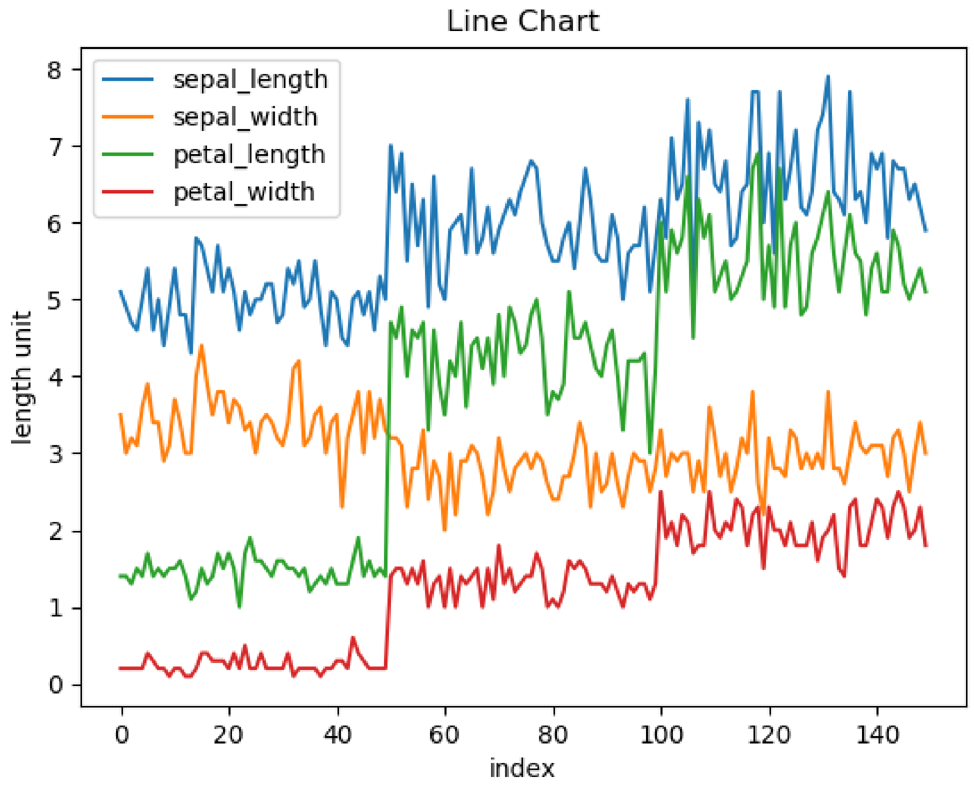


Try to change other parameters, such as range.

Line Chart

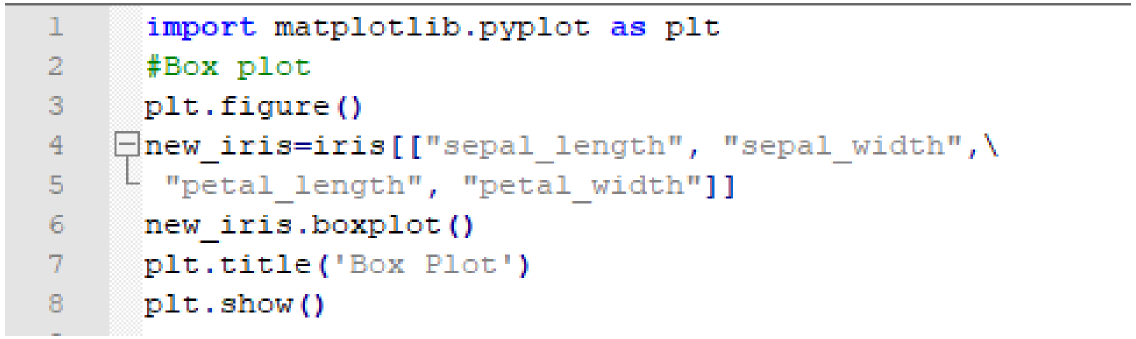


```
import matplotlib.pyplot as plt
#Line Chart
columns = iris.columns.drop(['class'])
# create x data
x_data = range(0, iris.shape[0])
# create figure and axis
fig, ax = plt.subplots()
# plot each column
for column in columns:
    ax.plot(x_data, iris[column])
# set title and legend
ax.set_title('Line Chart')
plt.xlabel('index')
plt.ylabel('length unit')
ax.legend(['sepal_length', 'sepal_width', 'petal_length', 'petal_width'])
plt.show()
```

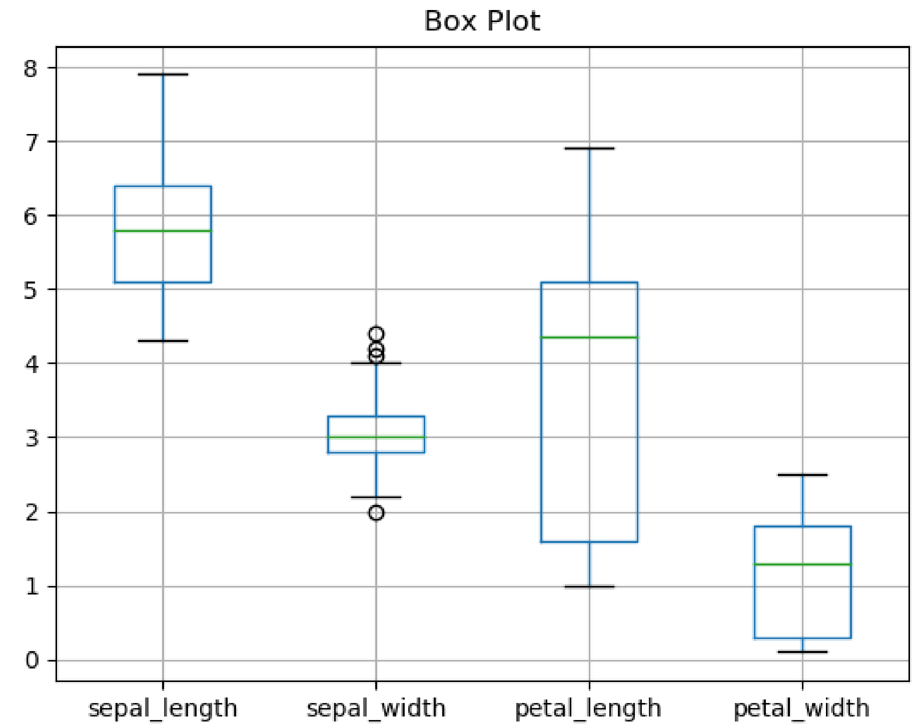


How to draw a Line Chart if we only want to plot sepal_length and sepal_width?

Box Plot



```
import matplotlib.pyplot as plt
#Box plot
plt.figure()
new_iris=iris[["sepal_length", "sepal_width",\
 "petal_length", "petal_width"]]
new_iris.boxplot()
plt.title('Box Plot')
plt.show()
```



Same question as Line Chart. if we only want to include sepal_length and sepal_width?