```
from sklearn.datasets import load_iris
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import pandas as pd
%matplotlib inline
```

```
data = load_iris()
dfx1 = pd.DataFrame(data.data, columns=data.feature_names)
dfy1 = pd.DataFrame(data.target, columns = ['class'])
df2 = dfx1.join(dfy1)
df2.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
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

```
columns = ['Sepal length', 'Sepal width', 'Petal length', 'Petal width', 'Class_labels']
dfx = pd.DataFrame(data.data, columns = data.feature_names)
dfy = pd.DataFrame(data.target, columns = ['class'])
df = dfx.join(dfy)e
df.head(10)
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
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
5	5.4	3.9	1.7	0.4	0
6	4.6	3.4	1.4	0.3	0
7	5.0	3.4	1.5	0.2	0
8	4.4	2.9	1.4	0.2	0
9	4.9	3.1	1.5	0.1	0

```
df.columns
```

```
Index(['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)',
       'petal width (cm)', 'class'],
      dtype='object')
```

```
df.describe()
```

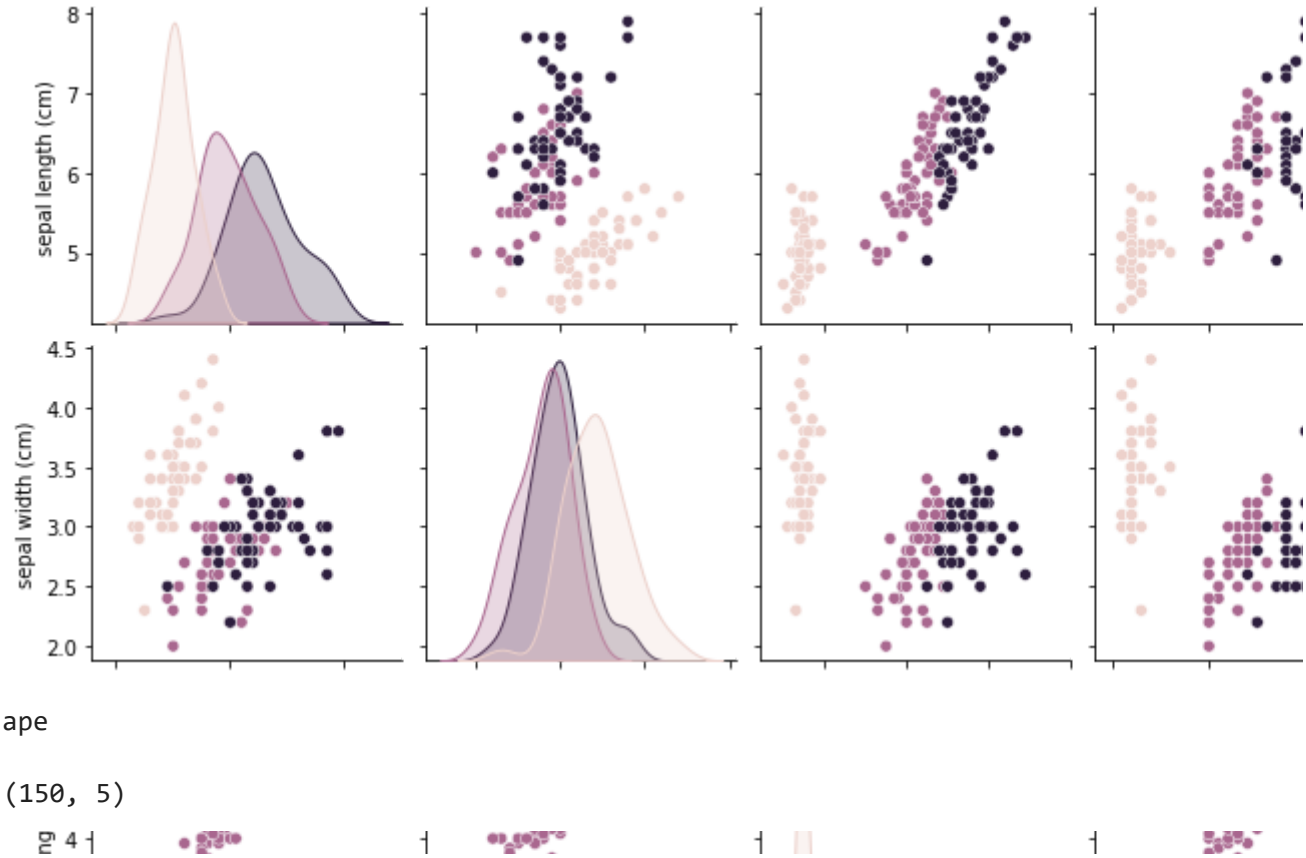
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
count	150.000000	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.057333	3.758000	1.199333	1.000000
std	0.828066	0.435866	1.765298	0.762238	0.819232
min	4.300000	2.000000	1.000000	0.100000	0.000000
25%	5.100000	2.800000	1.600000	0.300000	0.000000
50%	5.800000	3.000000	4.350000	1.300000	1.000000
75%	6.400000	3.300000	5.100000	1.800000	2.000000
max	7.900000	4.400000	6.900000	2.500000	2.000000

```
#Flower distribution
print(df.groupby('class').size())
```

```
class
0    50
1    50
2    50
dtype: int64
```

```
sns.pairplot(df, hue='class')     #visualize the dataset
```

```
<seaborn.axisgrid.PairGrid at 0x7f66c2b4b1d0>
```




```
df.shape
```

```
(150, 5)
```

```
df.isnull()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	class
0	False	False	False	False	False
1	False	False	False	False	False
2	False	False	False	False	False
3	False	False	False	False	False
4	False	False	False	False	False
...	...	...	...	...	...
145	False	False	False	False	False
146	False	False	False	False	False
147	False	False	False	False	False
148	False	False	False	False	False
149	False	False	False	False	False

150 rows × 5 columns

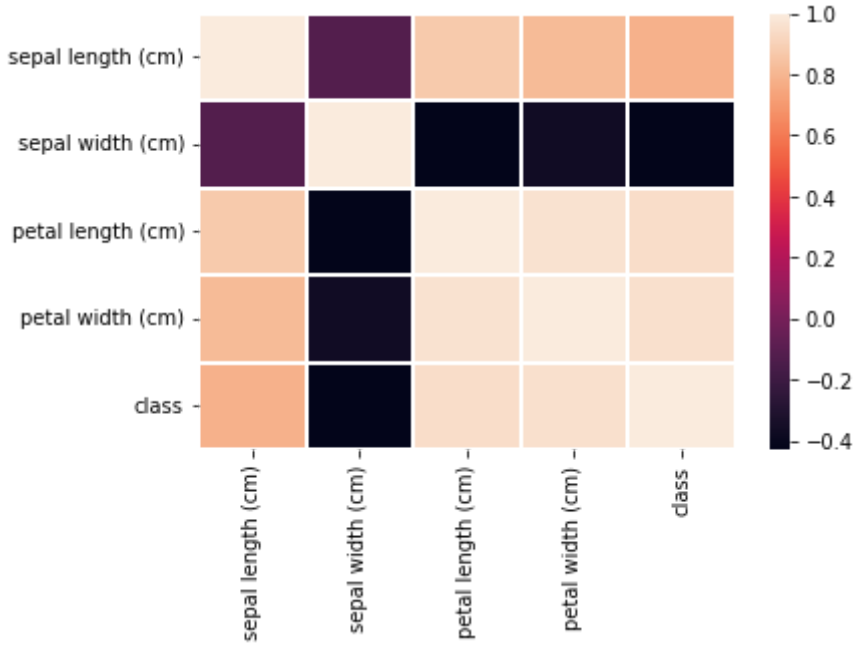
```
df.isnull().sum()
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
```

```
class                0
dtype: int64
```

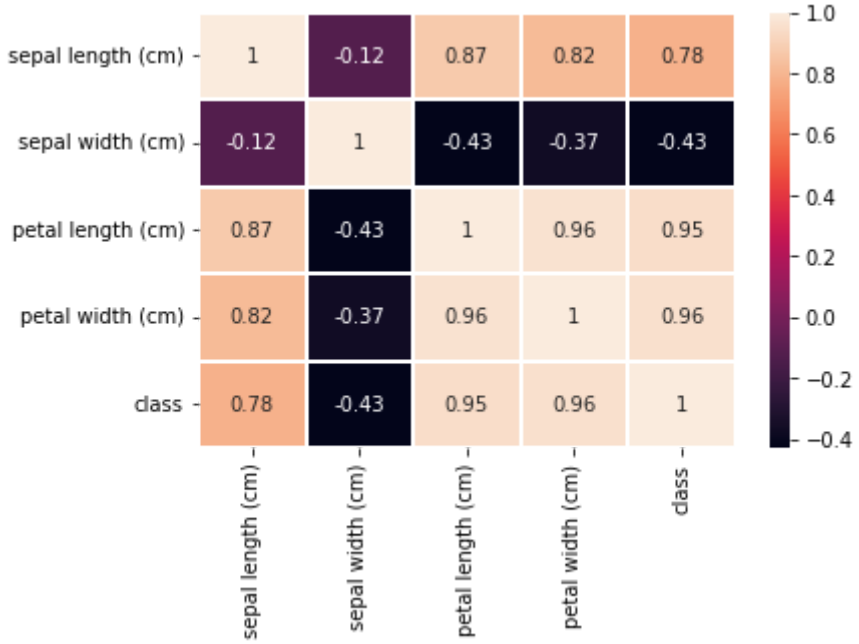
```
sns.heatmap(df.corr(),  linecolor = 'white', linewidths = 1)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66bc178510>
```




```
sns.heatmap(df.corr(),  linecolor = 'white', linewidths = 1, annot = True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f66bc101310>
```




```
df.corr(method='pearson')
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width
sepal length (cm)	1.000000	-0.117570	0.871754	0.8
sepal width (cm)	-0.117570	1.000000	-0.428440	-0.3
petal length (cm)	0.871754	-0.428440	1.000000	0.9

```
from sklearn import metrics
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split


X = df.drop(['class'], axis=1)
y = df['class']
# print(X.head())
print(X.shape)
# print(y.head())
print(y.shape)
```

```
(150, 4)
(150,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=5)
print(X_train.shape)
print(y_train.shape)
print(X_test.shape)
print(y_test.shape)
```

```
(90, 4)
(90,)
(60, 4)
(60,)
```

```
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print(metrics.accuracy_score(y_test, y_pred))
```

```
0.9833333333333333
/usr/local/lib/python3.7/dist-packages/sklearn/linear_model/_logistic.py:818: Converg
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG,
```

```
pred1 = logreg.predict([[6, 3, 4, 2]])
pred1
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451: UserWarning: X does not h
```

```
  "X does not have valid feature names, but"
array([1])
```

```
from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

```
                precision    recall  f1-score   support

             0       1.00      1.00      1.00        20
             1       1.00      0.95      0.98        21
             2       0.95      1.00      0.97        19

      accuracy                           0.98        60
     macro avg       0.98      0.98      0.98        60
  weighted avg       0.98      0.98      0.98        60
```