PS2

January 15, 2024

1 Iris plants classification

1.1 Load Iris dataset

[1]:

```
from sklearn.datasets import load_iris

iris = load_iris()
y = iris.target
X = iris.data
class_labels = iris.target_names
feature_names = iris.feature_names
print(X.shape)
print(y.shape)
print(class_labels)
print(feature_names)
```

```
(150, 4)
(150,)
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width
(cm)']
```

1.2 Data visualization

[2]:

```
print(y==0)
```

```
[ True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True  True  True  True  True  True  True  True  True  True  True
  True  True False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False False False False False False False
 False False False False False False]
```

[3]:

```
print(X[y==0]) # extract setosa's features
```

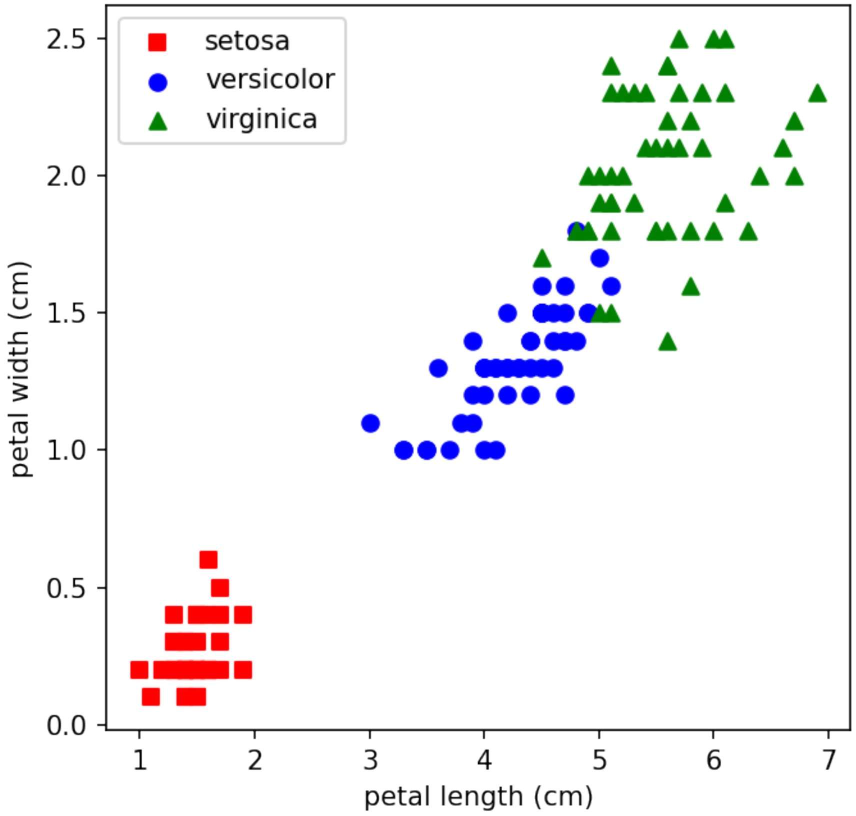
```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]
 [5.4 3.9 1.7 0.4]
 [4.6 3.4 1.4 0.3]
 [5.  3.4 1.5 0.2]
 [4.4 2.9 1.4 0.2]
 [4.9 3.1 1.5 0.1]
 [5.4 3.7 1.5 0.2]
 [4.8 3.4 1.6 0.2]
 [4.8 3.  1.4 0.1]
 [4.3 3.  1.1 0.1]
 [5.8 4.  1.2 0.2]
 [5.7 4.4 1.5 0.4]
 [5.4 3.9 1.3 0.4]
 [5.1 3.5 1.4 0.3]
 [5.7 3.8 1.7 0.3]
 [5.1 3.8 1.5 0.3]
 [5.4 3.4 1.7 0.2]
 [5.1 3.7 1.5 0.4]
 [4.6 3.6 1.  0.2]
 [5.1 3.3 1.7 0.5]
 [4.8 3.4 1.9 0.2]
 [5.  3.  1.6 0.2]
 [5.  3.4 1.6 0.4]
 [5.2 3.5 1.5 0.2]
 [5.2 3.4 1.4 0.2]
 [4.7 3.2 1.6 0.2]
 [4.8 3.1 1.6 0.2]
 [5.4 3.4 1.5 0.4]
 [5.2 4.1 1.5 0.1]
 [5.5 4.2 1.4 0.2]
 [4.9 3.1 1.5 0.2]
 [5.  3.2 1.2 0.2]
 [5.5 3.5 1.3 0.2]
 [4.9 3.6 1.4 0.1]
 [4.4 3.  1.3 0.2]
 [5.1 3.4 1.5 0.2]
 [5.  3.5 1.3 0.3]
 [4.5 2.3 1.3 0.3]
 [4.4 3.2 1.3 0.2]
 [5.  3.5 1.6 0.6]
 [5.1 3.8 1.9 0.4]
 [4.8 3.  1.4 0.3]
 [5.1 3.8 1.6 0.2]
 [4.6 3.2 1.4 0.2]
 [5.3 3.7 1.5 0.2]
 [5.  3.3 1.4 0.2]]
```

[4]:

```
import matplotlib.pyplot as plt

X_y0 = X[y==0] # setosa
X_y1 = X[y==1] # versicolor
X_y2 = X[y==2] # virginica

plt.figure(figsize=(5,5), dpi=150)
plt.scatter (X_y0[:,2], X_y0[:,3],
            c='red', label='setosa',marker='s')
plt.scatter (X_y1[:,2], X_y1[:,3],
            c='blue', label='versicolor',marker='o')
plt.scatter (X_y2[:,2], X_y2[:,3],
            c='green', label='virginica',marker='^')
plt.xlabel(iris.feature_names[2])
plt.ylabel(iris.feature_names[3])
plt.legend()
plt.show()
```



1.3 Train-test data split

[5]:

```
from sklearn.model_selection import train_test_split

X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2)
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(120, 4)
(30, 4)
(120,)
(30,)
```

1.4 Least squares

[6]:

```
from sklearn.linear_model import RidgeClassifier

Model_LS = RidgeClassifier()

# training
Model_LS.fit(X_train,y_train)

# prediction on test data
y_pred = Model_LS.predict(X_test)
print(y_pred)
print(y_test)

# evaluate test accuracy
test_accuracy = Model_LS.score(X_test,y_test)
print(test_accuracy)
```

```
[2 0 1 2 1 0 0 0 0 1 1 2 0 0 1 1 1 2 2 2 1 2 1 2 1 1 0 1 0 2]
[1 0 2 2 1 0 0 0 0 1 1 2 0 0 2 1 2 2 1 2 1 2 1 1 1 1 0 1 0 2]
0.8
```

1.5 Logistic regression

[7]:

```
from sklearn.linear_model import LogisticRegression

Model_LR = LogisticRegression()

# training
Model_LR.fit(X_train,y_train)
```

```python
# prediction on test data
y_pred = Model_LR.predict(X_test)
print(y_pred)
print(y_test)

# evaluate test accuracy
test_accuracy = Model_LR.score(X_test,y_test)
print(test_accuracy)
```

```
[1 0 1 1 1 0 0 0 0 1 1 2 0 0 2 1 1 2 1 2 1 2 1 1 1 1 0 1 0 2]
[1 0 2 2 1 0 0 0 0 1 1 2 0 0 2 1 2 2 1 2 1 2 1 1 1 1 0 1 0 2]
0.9
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\sklearn\linear_model\_logistic.py:814: ConvergenceWarning: lbfgs failed
to converge (status=1):
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
  n_iter_i = _check_optimize_result(
```

[ ]: