SCIKIT-LEARN MECHINE LEARNING IN PYTHON

Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

SCIKIT-LEARN

- Simple and efficient tools for predictive data analysis
- Accessible to everybody, and reusable in various contexts
- Built on NumPy, SciPy, and matplotlib
- Open source, commercially usable - BSD license

INSTALLATION OF SCIKIT-LEARN

Dependencies

scikit-learn requires:

- Python (>= 3.6)
- NumPy (>= 1.13.3)
- SciPy (>= 0.19.1)
- joblib (>= 0.11)
- threadpoolctl (>= 2.0.0)

Commands:

In normal python distribution:

```
pip install -U scikit-learn
```

In order to check your installation you can use

```
python -m pip show scikit-learn  # to see which version and where scikit-learn is installed
python -m pip freeze  # to see all packages installed in the active virtualenv
python -c "import sklearn; sklearn.show_versions()"
```

In Anaconda Environment:

```
conda install -c conda-forge scikit-learn
conda install -c conda-forge scikit-learn
```

In order to check your installation you can use

```
conda list scikit-learn  # to see which scikit-learn version is installed
conda list  # to see all packages installed in the active conda environment
python -c "import sklearn; sklearn.show_versions()"
```

CLASSIFICATION

Identifying which category an object belongs to.

Applications: Spam detection, image recognition.

Algorithms: SVM, nearest neighbors, random forest, etc.

REGRESSION

Predicting a continuous-valued attribute associated with an object.

Applications: Drug response, Stock prices.

Algorithms: SVR, nearest neighbors, random forest, etc.

IRIS DATA

The data set consists of 50 samples from three species of iris- iris Setosa,Virginica and Versicolor

Four features were measured from each sample : Leangth and the width of the sepals and petals, in cetimeters

```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving irisdataset.csv to irisdataset.csv
```

```
import pandas as pd
import numpy as np
dataset=pd.read_csv('irisdataset.csv')
dataset
```

```
x=dataset.iloc[:,0:4].values
```

```
y=dataset.iloc[:,4].values
```

```
from sklearn.preprocessing import LabelEncoder
labelencoder_y=LabelEncoder()
y=labelencoder_y.fit_transform(y)
```

```
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)
```

```
from sklearn.linear_model import LogisticRegression
logmodel=LogisticRegression()
logmodel.fit(x_train,y_train)
```

```
y_pred=logmodel.predict(x_test)
y_pred
```

```
     array([0, 1, 0, 2, 0, 2, 1, 2, 2, 0, 2, 1, 2, 0, 2, 0, 0, 0, 1, 2, 0, 2,
            1, 1, 0, 0, 0, 1, 0, 0])
```

```
y_test
```

```
     array([0, 1, 0, 2, 0, 2, 1, 2, 2, 0, 2, 1, 2, 0, 2, 0, 0, 0, 1, 2, 0, 2,
            1, 1, 0, 0, 0, 1, 0, 0])
```

```
from sklearn.metrics import confusion_matrix
confusion_matrix(y_test,y_pred)
```

```
     array([[14,  0,  0],
            [ 0,  7,  0],
            [ 0,  0,  9]])
```

```
30/30
```

```
     1.0
```

```
from sklearn.neighbors import KNeighborsClassifier
classifier_knn= KNeighborsClassifier(n_neighbors=5,metric='minkowski',p=2)
classifier_knn.fit(x_train,y_train)
```

```
     KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                          metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                          weights='uniform')
```

```
y_pred=classifier_knn.predict(x_test)
```

```
confusion_matrix(y_test,y_pred)
```

```
     array([[14,  0,  0],
            [ 0,  6,  1],
            [ 0,  0,  9]])
```

```
29/30
```

```
     0.9666666666666667
```

```
from sklearn.naive_bayes import GaussianNB
classifier_nb=GaussianNB()
classifier_nb.fit(x_train,y_train)
```

```
     GaussianNB(priors=None, var_smoothing=1e-09)
```

```
y_pred=classifier_nb.predict(x_test)
confusion_matrix(y_test,y_pred)
```

```
     array([[14,  0,  0],
            [ 0,  7,  0],
            [ 0,  1,  8]])
```

```
29/30
```

```
     0.9666666666666667
```

```
from sklearn.svm import SVC
classifier_svm_sigmoid=SVC(kernel='sigmoid')
classifier_svm_sigmoid.fit(x_train,y_train)
```

```
     SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
         decision_function_shape='ovr', degree=3, gamma='scale', kernel='sigmoid',
         max_iter=-1, probability=False, random_state=None, shrinking=True,
         tol=0.001, verbose=False)
```

```
y_pred=classifier_svm_sigmoid.predict(x_test)
confusion_matrix(y_test,y_pred)
```

```
     array([[ 0, 14,  0],
            [ 0,  7,  0],
            [ 0,  9,  0]])
```

```
7/30
```

```
     0.23333333333333334
```

```
from sklearn.svm import SVC
```

```
classifier_svm_linear=SVC(kernel='linear')
classifier_svm_linear.fit(x_train,y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='linear',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred=classifier_svm_linear.predict(x_test)
confusion_matrix(y_test,y_pred)
```

```
array([[14,  0,  0],
       [ 0,  7,  0],
       [ 0,  0,  9]])
```

```
30/30
```

```
1.0
```

```
from sklearn.svm import SVC
classifier_svm_rbf=SVC(kernel='rbf')
classifier_svm_rbf.fit(x_train,y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='rbf',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred=classifier_svm_rbf.predict(x_test)
confusion_matrix(y_test,y_pred)
```

```
array([[14,  0,  0],
       [ 0,  7,  0],
       [ 0,  0,  9]])
```

```
30/30
```

```
1.0
```

```
from sklearn.svm import SVC
classifier_svm_poly=SVC(kernel='poly')
classifier_svm_poly.fit(x_train,y_train)
```

```
SVC(C=1.0, break_ties=False, cache_size=200, class_weight=None, coef0=0.0,
    decision_function_shape='ovr', degree=3, gamma='scale', kernel='poly',
    max_iter=-1, probability=False, random_state=None, shrinking=True,
    tol=0.001, verbose=False)
```

```
y_pred=classifier_svm_poly.predict(x_test)
confusion_matrix(y_test,y_pred)
confusion_matrix(y_test,y_pred)
```

```
array([[14,  0,  0],
       [ 0,  6,  1],
       [ 0,  0,  9]])
```

```
29/30
```

```
0.9666666666666667
```

```
from sklearn.tree import DecisionTreeClassifier
classifier_dt=DecisionTreeClassifier(criterion='entropy')
classifier_dt.fit(x_train,y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
y_pred=classifier_dt.predict(x_test)
confusion_matrix(y_test,y_pred)
```

```
array([[14,  0,  0],
       [ 0,  6,  1],
       [ 0,  1,  8]])
```

```
28/30
```

```
0.9333333333333333
```

```
from sklearn.ensemble import RandomForestClassifier
classifier_rf=RandomForestClassifier(n_estimators=3,criterion='entropy')
classifier_rf.fit(x_train,y_train)
```

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='entropy', max_depth=None, max_features='auto',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=3,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

```
y_pred=classifier_rf.predict(x_test)
confusion_matrix(y_test,y_pred)
```

```
array([[14,  0,  0],
       [ 0,  6,  1],
       [ 0,  1,  8]])
```

```
28/30
```

```
0.9333333333333333
```