223681886 - Kaushalya Rajapaksha - 3.1P - SIT725

Scikit-learn embodies the essence of machine learning in Python, providing intuitive and effective tools for predictive data analysis. Accessible to all users, its versatility spans diverse applications, leveraging the robust foundations of NumPy, SciPy, and matplotlib. As an open-source platform under the BSD license, it ensures both accessibility and commercial viability.

Loading datasets

Scikit-learn facilitates model development by providing two sample datasets, sparing users the complexities of sourcing external data. This feature, available through the sklearn.datasets module, streamlines the initiation of machine learning projects, eliminating the need for laborious dataset acquisition processes.

sklearn preprocessing helps to transformer classes to change raw feature vectors.
using **panda library** able to get a dataframe with rows and columns out of the diabetes dataset.

In [112…

```
from sklearn import preprocessing
import pandas as pd
```

Example 01

import **diabetes dataset** from the sklearn library.
sklearn datasets are pre-processed and ready to use and **load the diabetes** dataset.
this data set available with information on 442 patient

In [113…

```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
```

Example 02

import **iris dataset** from the sklearn library.
this data set available with information on 150 measurements

In [114…

```
from sklearn.datasets import load_iris
iris = load_iris()
```

Show Information

In here user can get the idea about dataset information, values, shape and lenth

Example 01

In [115…

```
print(diabetes.DESCR)
```

```
.. _diabetes_dataset:

Diabetes dataset
----------------

Ten baseline variables, age, sex, body mass index, average blood
pressure, and six blood serum measurements were obtained for each of n =
442 diabetes patients, as well as the response of interest, a
quantitative measure of disease progression one year after baseline.

**Data Set Characteristics:**

  :Number of Instances: 442

  :Number of Attributes: First 10 columns are numeric predictive values

  :Target: Column 11 is a quantitative measure of disease progression one year after baseline

  :Attribute Information:
      - age     age in years
      - sex
      - bmi     body mass index
      - bp      average blood pressure
      - s1      tc, total serum cholesterol
      - s2      ldl, low-density lipoproteins
      - s3      hdl, high-density lipoproteins
      - s4      tch, total cholesterol / HDL
      - s5      ltg, possibly log of serum triglycerides level
      - s6      glu, blood sugar level

Note: Each of these 10 feature variables have been mean centered and scaled by the standard deviation times the square root of `n_samples` (i.e. the sum of squares of each column totals 1).

Source URL:
https://www4.stat.ncsu.edu/~boos/var.select/diabetes.html

For more information see:
Bradley Efron, Trevor Hastie, Iain Johnstone and Robert Tibshirani (2004) "Least Angle Regression," Annals of Statistics (with discussion), 407-499.
(https://web.stanford.edu/~hastie/Papers/LARS/LeastAngle_2002.pdf)
```

create the **dataframe** from diabetes dataset and add **columns name** to dataset and add diabetes **dataset target** as column of **measure**
show the first 10 rows detailed information of each and every attributes of the dataset

DESCR can display the description of dataset, shape, number of rows, columns and attribute information.

In [116…

```
df = pd.DataFrame(diabetes.data, columns = diabetes.feature_names)
df['measure'] = diabetes.target
df.head(10)
```

Out[116…

	age	sex	bmi	bp	s1	s2	s3	s4	s5	s6	measure
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592	0.019907	-0.017646	151.0
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493	-0.068332	-0.092204	75.0
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592	0.002861	-0.025930	141.0
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309	0.022688	-0.009362	206.0
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592	-0.031988	-0.046641	135.0
5	-0.092695	-0.044642	-0.040696	-0.019442	-0.068991	-0.079288	0.041277	-0.076395	-0.041176	-0.096346	97.0
6	-0.045472	0.050680	-0.047163	-0.015999	-0.040096	-0.024800	0.000779	-0.039493	-0.062917	-0.038357	138.0
7	0.063504	0.050680	-0.001895	0.066629	0.090620	0.108914	0.022869	0.017703	-0.035816	0.003064	63.0
8	0.041708	0.050680	0.061696	-0.040099	-0.013953	0.006202	-0.028674	-0.002592	-0.014960	0.011349	110.0
9	-0.070900	-0.044642	0.039062	-0.033213	-0.012577	-0.034508	-0.024993	-0.002592	0.067737	-0.013504	310.0

check for existing null values

In [117…

```python
total_null_values = sum(df.isnull().sum())
print(total_null_values)
```

```
0
```

count the number of zero values in each column

In [118…

```python
print((df[['age','sex','bmi','bp', 's1', 's2', 's3', 's4', 's5', 's6', 'measure']] == 0).sum())
```

```
age        0
sex        0
bmi        0
bp         0
s1         0
s2         0
s3         0
s4         0
s5         0
s6         0
measure    0
dtype: int64
```

Example 02

In [119…

```python
print(iris.DESCR)
```

```
.. _iris_dataset:

Iris plants dataset
--------------------

**Data Set Characteristics:**

    :Number of Instances: 150 (50 in each of three classes)
    :Number of Attributes: 4 numeric, predictive attributes and the class
    :Attribute Information:
        - sepal length in cm
        - sepal width in cm
        - petal length in cm
        - petal width in cm
        - class:
                - Iris-Setosa
                - Iris-Versicolour
                - Iris-Virginica

    :Summary Statistics:

    ============== ==== ==== ======= ===== ====================
                    Min  Max   Mean    SD   Class Correlation
    ============== ==== ==== ======= ===== ====================
    sepal length:   4.3  7.9   5.84   0.83    0.7826
    sepal width:    2.0  4.4   3.05   0.43   -0.4194
    petal length:   1.0  6.9   3.76   1.76    0.9490  (high!)
    petal width:    0.1  2.5   1.20   0.76    0.9565  (high!)
    ============== ==== ==== ======= ===== ====================

    :Missing Attribute Values: None
    :Class Distribution: 33.3% for each of 3 classes.
    :Creator: R.A. Fisher
    :Donor: Michael Marshall (MARSHALL%PLU@io.arc.nasa.gov)
    :Date: July, 1988

The famous Iris database, first used by Sir R.A. Fisher. The dataset is taken
from Fisher's paper. Note that it's the same as in R, but not as in the UCI
Machine Learning Repository, which has two wrong data points.

This is perhaps the best known database to be found in the
pattern recognition literature.  Fisher's paper is a classic in the field and
is referenced frequently to this day.  (See Duda & Hart, for example.)  The
data set contains 3 classes of 50 instances each, where each class refers to a
type of iris plant.  One class is linearly separable from the other 2; the
latter are NOT linearly separable from each other.

.. topic:: References

   - Fisher, R.A. "The use of multiple measurements in taxonomic problems"
     Annual Eugenics, 7, Part II, 179-188 (1936); also in "Contributions to
     Mathematical Statistics" (John Wiley, NY, 1950).
   - Duda, R.O., & Hart, P.E. (1973) Pattern Classification and Scene Analysis.
     (Q327.D83) John Wiley & Sons.  ISBN 0-471-22361-1.  See page 218.
   - Dasarathy, B.V. (1980) "Nosing Around the Neighborhood: A New System
     Structure and Classification Rule for Recognition in Partially Exposed
     Environments".  IEEE Transactions on Pattern Analysis and Machine
     Intelligence, Vol. PAMI-2, No. 1, 67-71.
   - Gates, G.W. (1972) "The Reduced Nearest Neighbor Rule".  IEEE Transactions
     on Information Theory, May 1972, 431-433.
   - See also: 1988 MLC Proceedings, 54-64.  Cheeseman et al"s AUTOCLASS II
     conceptual clustering system finds 3 classes in the data.
   - Many, many more ...
```

create the **dataframe** from iris dataset and add columns name to dataset

show the first 10 rows detailed information of each and every attributes of the dataset

In [120…

```python
df1 = pd.DataFrame(iris.data, columns = iris.feature_names)
df1['measure'] = iris.target
df1.head(10)
```

Out[120…

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)	measure
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

check for existing null values

In [121…

```python
total_null_values1 = sum(df1.isnull().sum())
print(total_null_values1)
```

```
0
```

count the number of zero values in each column

In [122…

```python
print((df1[['sepal length (cm)','sepal width (cm)','petal length (cm)','petal width (cm)']] == 0).sum())
```

```
sepal length (cm)    0
sepal width (cm)     0
petal length (cm)    0
petal width (cm)     0
dtype: int64
```

Preprocessing, RandomForestClassifier and Metrics

Example 01

dataset separated to 80/20 for training and testing

standardize and train/test split of diabetes dataset.

split diabetes dataset to train and test in to random number of 20% testing and 80% training and test size should be between 0.0 and 1.0

In [123…

```python
from sklearn.model_selection import train_test_split
diabetes.data = preprocessing.scale(diabetes.data)

X_train, X_test, Y_train, Y_test = train_test_split(
    diabetes.data, diabetes.target, test_size=0.2, random_state=5)
```

display the traing and testing dataser label and feature details

In [124…

```python
print(X_train.shape,' is the shape of Training Data Features')
print(Y_train.shape,' is the shape of Training Data Lables')
print(X_test.shape,' is the shape of Testing Data Features')
print(Y_test.shape,' is the shape of Testing Data Lables')
```

```
(353, 10)  is the shape of Training Data Features
(353,)  is the shape of Training Data Lables
(89, 10)  is the shape of Testing Data Features
(89,)  is the shape of Testing Data Lables
```

impliment the **Random Forest Classifier Algorithm** and import **RandomForestClassifier** from sklearn

sklearn.metrics includes score functions and performance metrics and pairwise metrics and distance computations

calculate test data scores over 10, 100, 200, 500 and 1000 trees

Random Forest Classifier Algorithm

- Allows the machine or software agent to learn its behavior based on feedback from the environment.
- This behavior can be learnt once and for all, or keep on adapting as time goes by.
- A reward function is used to measure the reward for a given action

In [125…
```python
from sklearn.ensemble import RandomForestClassifier
import sklearn.metrics as sm
rf_scores = []
estimators = [10, 100, 200, 500, 1000]
for i in estimators:
    rf_classifier = RandomForestClassifier(n_estimators=i, random_state = 42)
    rf_classifier.fit(X_train, Y_train)
    rf_scores.append(rf_classifier.score(X_test, Y_test))
```

predictions using test data and get the **accuracy score** for dataset

printing the **accuaracy score** for the test data

In [126…
```python
Y_pred = rf_classifier.predict(X_test)
Y_pred
accuracy_score=sm.accuracy_score(Y_test, Y_pred)
print('Accuracy score given for test data:',str(accuracy_score))
```

```
Accuracy score given for test data: 0.02247191011235955
```

Example 02

dataset separated to 75/25 for training and testing standardize and train/test split of diabetes dataset. split diabetes dataset to train and test in to random number of 25% testing and 75% training and test size should be between 0.0 and 1.0

In [127…
```python
from sklearn.model_selection import train_test_split
iris.data = preprocessing.scale(iris.data)

X_train1, X_test1, Y_train1, Y_test1 = train_test_split(
    iris.data, iris.target, test_size=0.25, random_state=5)
```

display the traing and testing dataser label and feature details

In [128…
```python
print(X_train1.shape,' is the shape of Training Data Features')
print(Y_train1.shape,' is the shape of Training Data Lables')
print(X_test1.shape,' is the shape of Testing Data Features')
print(Y_test1.shape,' is the shape of Testing Data Lables')
```

```
(112, 4)  is the shape of Training Data Features
(112,)  is the shape of Training Data Lables
(38, 4)  is the shape of Testing Data Features
(38,)  is the shape of Testing Data Lables
```

impliment the **Support Vector Machines** and import **SVM** from sklearn

Support Vector Machines Algorithm (SVM)

- When the correct classes (labels) of the training data are known we can use Supervised Learning (Classification).

In [129…
```python
from sklearn import svm
from sklearn.svm import SVC
svm_classiifier = SVC(kernel='linear', C=1.0, random_state=42)
svm_classiifier.fit(X_train1, Y_train1)
Y_pred1 = svm_classiifier.predict(X_test1)
```

predictions using test data and get the **accuracy score** for dataset

printing the **accuaracy score** for the test data

In [130…
```python
accuracy_score1 = sm.accuracy_score(Y_test1, Y_pred1)
print('Accuracy score given for test data:',str(accuracy_score1))
```

```
Accuracy score given for test data: 0.9210526315789473
```

Visualisation

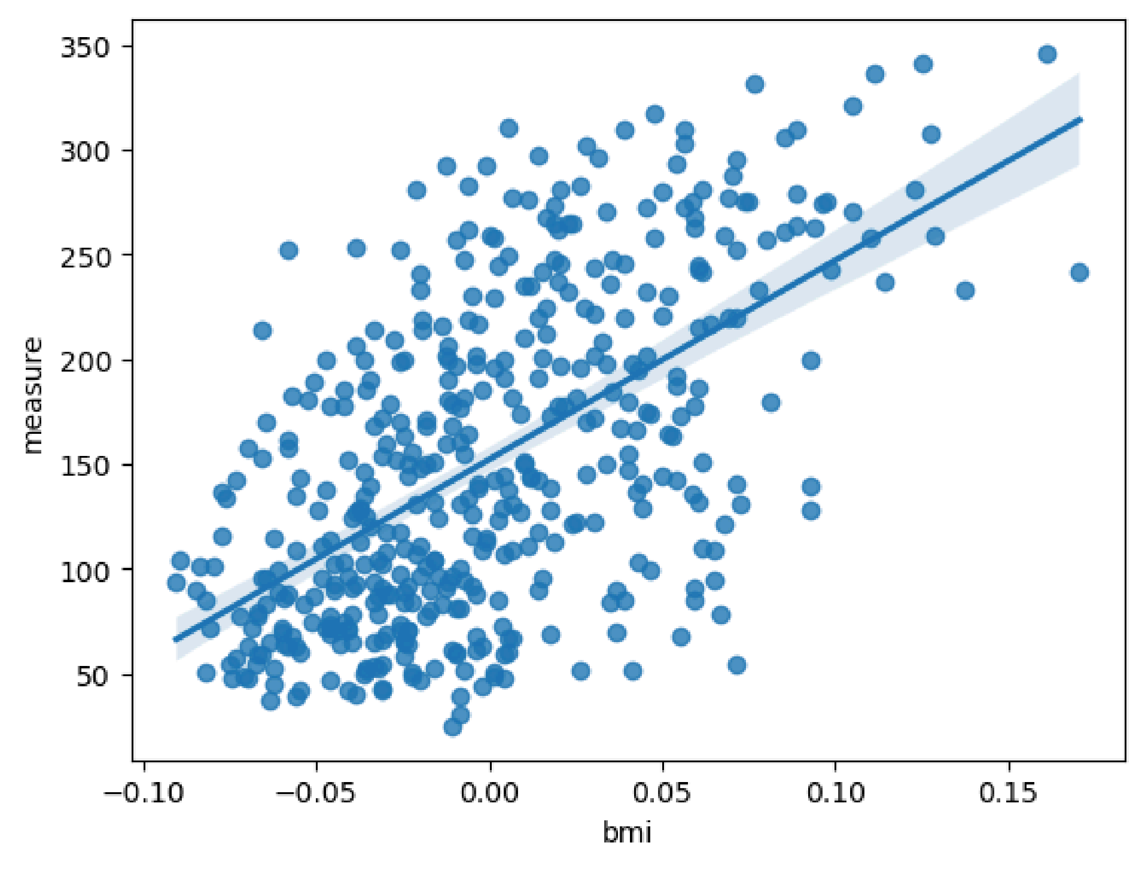
Example 01

import statistical graphics library to show graphical details of diabetes bmi and measure(target)

set the fields to visualisation

In [131…
```python
import seaborn as sns
sns.regplot(x="bmi", y="measure", data=df)
```

Out[131…
```
<Axes: xlabel='bmi', ylabel='measure'>
```

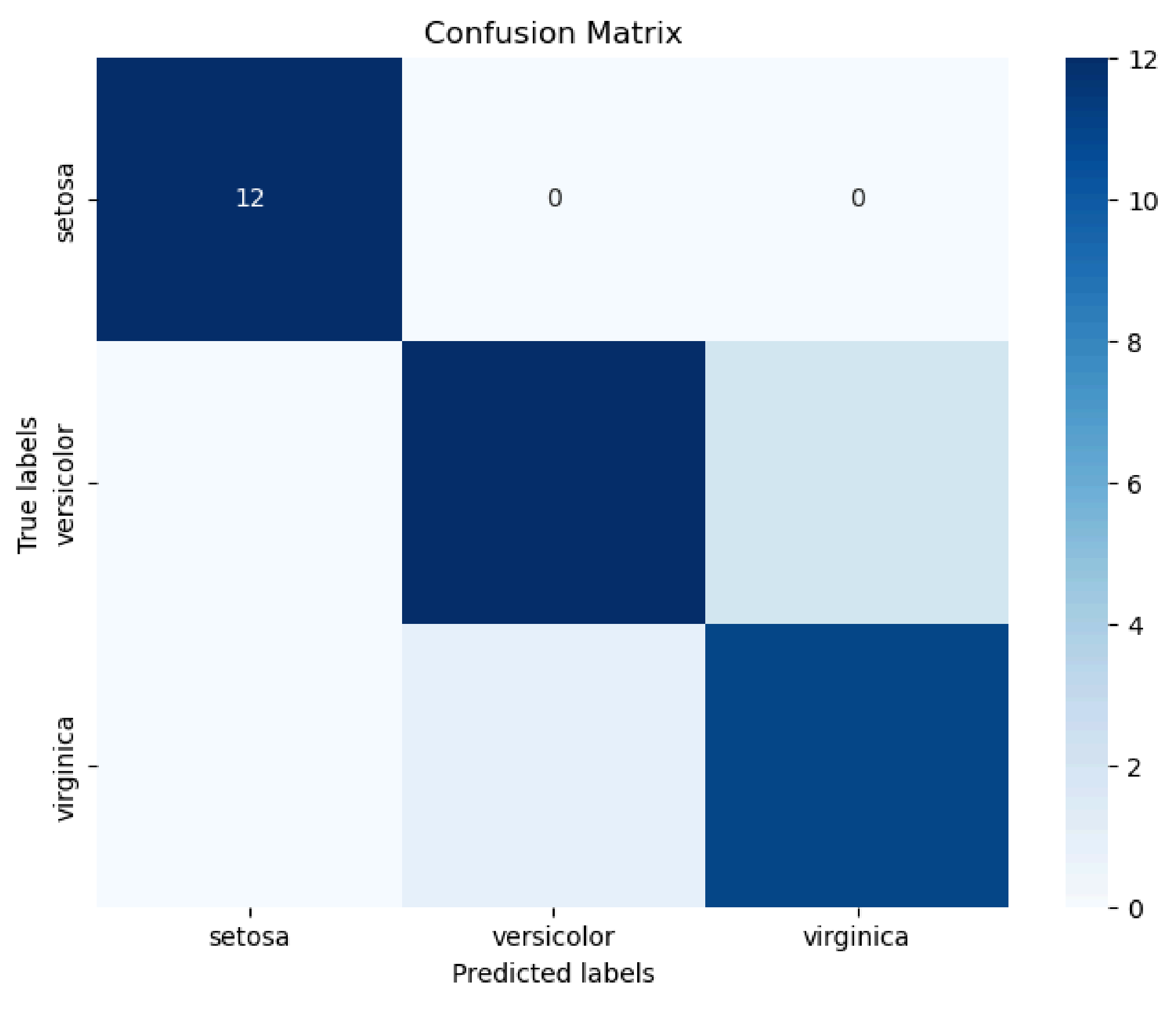


Example 02

generates confusion matrix based on predicted outputs and real labels

In [132…
```python
from sklearn.metrics import confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns;

con_metrix = confusion_matrix(Y_test1, Y_pred1)
plt.figure(figsize=(8,6))
sns.heatmap(con_metrix, annot=True, cmap='Blues', fmt='g', xticklabels=iris.target_names, yticklabels=iris.target_names)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title('Confusion Matrix')
plt.show()
```


Confusion Matrix
setosa
versicolor
virginica
True labels
setosa
versicolor
virginica
Predicted labels