EXPERIMENT NUMBER : 5
EXPERIMENT NAME: MACHINE LEARNING USING IRIS DATASET
DATE : 17/11/2022, THURSDAY

* AIM:

To implement various machine learning techniques using Raspberry Pi.

* INTEGRATED DEVELOPMENT ENVIRONMENT (IDE):

Name - Thonny 4.0.1 Python 3.10.6
Publisher - Aivar Annamaa
Support link - https://thonny.org

* IMPORT NECESSARY LIBRARIES:

```
import sys #Provides access to some variables used or maintained
           by the interpreter and to functions that interact
           strongly with the interpreter.
print ('Python : {}'.format(sys.version))

import scipy #Includes modules for statistics, optimization,
             integration, linear algebra, Fourier transforms,
             signal and image processing, and more
print ('scipy : {}'.format(scipy.__version__))

import numpy #support for large, multi-dimensional arrays
             and matrices
print ('numpy : {}'.format(numpy.__version__))

import matplotlib #Provides a MATLAB-like plotting framework
print ('matplotlib : {}'.format(matplotlib.__version__))

import pandas #Library for working with data sets
print ('pandas : {}'.format(pandas.__version__))
```

import sklearn # Provides various tools for model fitting, data preprocessing, model selection, model evaluation, and many other utilities
print ('sklearn : {}'. format (sklearn. __version__)

(a) Univariate and Multivariate Data Plots using IRIS dataset :

→ Algorithm -

① Import pandas, library for working with data sets.
② From pandas. plotting import scatter_matrix, to draw a matrix of scatter plots.
③ Import matplotlib. pyplot as plt, collection of command style functions that make matplotlib work like MATLAB.
④ Load the dataset and read comma-seperated values (csv) file into DataFrame.
⑤ Return a tuple representing the dimensionality of the DataFrame and return the first n rows.
⑥ Generate descriptive statistics that summarizes the central tendency, dispersion and shape of a dataset's distribution, excluding NaN values.
⑦ Group DataFrame using a mapper or by a Series of columns and make plots of Series or DataFrame.
⑧ Make a histogram of the DataFrame's columns and visualize data with scatter plots.

→ Python Code -

```
import pandas
from pandas. plotting import scatter_matrix
import matplotlib. pyplot as plt

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/
      master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length',
         'petal-width', 'class']
dataset = pandas. read_csv (url, names = names)
```

```
print ('Dataset Dimensions: ' + str(dataset.shape))
print ('\nHead of the Data: \n' + str(dataset.head(20)))
print ('\nData statistics: \n' + str(dataset.describe))
print ('\nClass Distribution: \n' + str(dataset.groupby('class').size()))

dataset.plot(kind='box', layout=(2,2), sharex=False, sharey=
False); plt.show().
dataset.hist(); plt.show()
scatter_matrix(dataset); plt.show()
```

(b) Analysis of Machine Learning using IRIS Dataset:

→ Algorithm -

① Import matplotlib.pyplot as plt, a collection of command style functions that make matplotlib work like MATLAB.

② Import pandas, library for working with data sets. Import scatter_matrix from pandas.plotting to draw a matrix of scatter plots.

③ From sklearn import model_selection, split arrays or matrices into random train and test subsets.

④ Import metrics for classification technique and other model building libraries.

⑤ Load dataset and read the comma-seperated values (csv) file into DataFrame.

⑥ Create training and validation datasets and return a Numpy representation of the DataFrame. Assume independent (data) variable and dependent (target) variable.

⑦ Control the shuffling applied to the data before applying the split and set the scoring criteria.

⑧ Build all the models first, and then evaluate each model.

⑨ Provide train/test indices to split data into train/test sets and evaluate a score by cross-validation.

PTO

Dataset Dimension
(150, 5)
Head of Data :

	sepal-length	sepal-width	petal-length	petal-width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa
15	5.7	4.4	1.5	0.4	Iris-setosa
16	5.4	3.9	1.3	0.4	Iris-setosa
17	5.1	3.5	1.4	0.3	Iris-setosa
18	5.7	3.8	1.7	0.3	Iris-setosa
19	5.1	3.8	1.5	0.3	Iris-setosa

statistics

	sepal-length	sepal-width	petal-length	petal-width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

Class Distribution

class	
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

dtype: int64

Figure 1 - Descriptive Statistics of IRIS

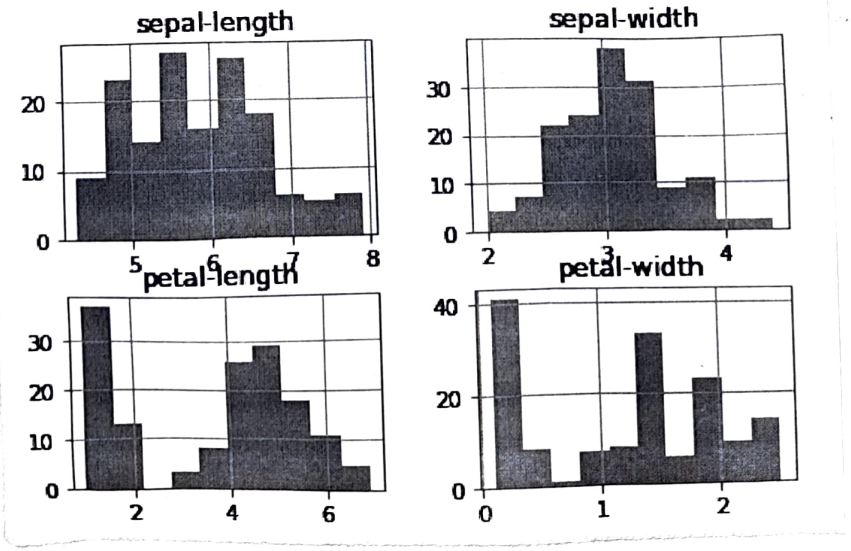

Figure 2 - Histogram of IRIS columns

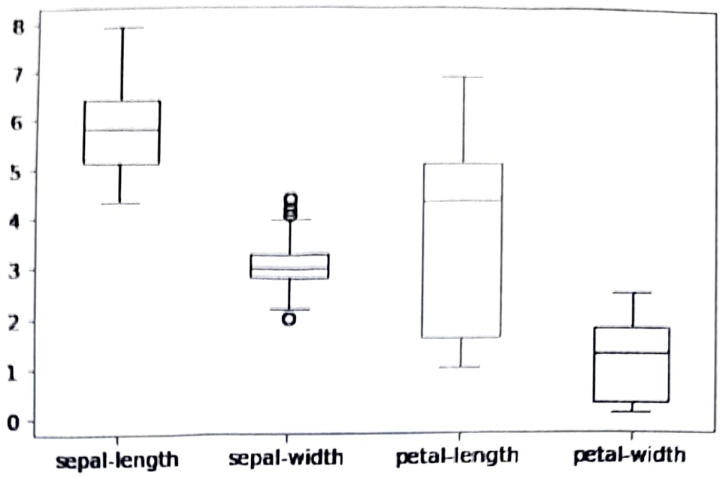

Figure 3 - Box Plot

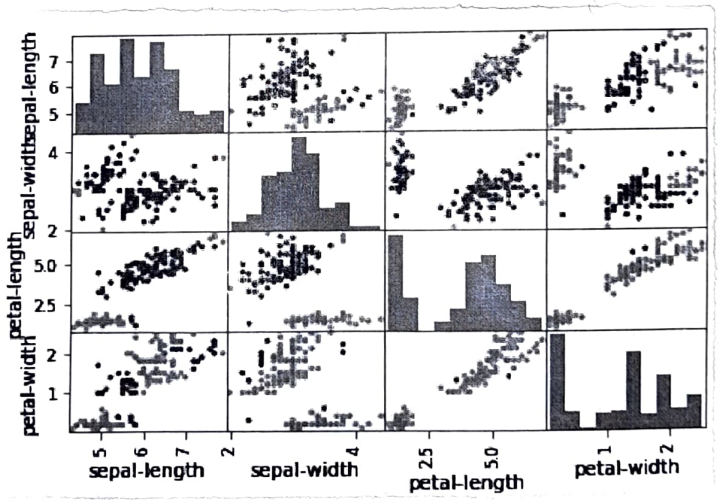

Figure 4 - Matrix of scatter plots

→ Python Code -

```
import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix
from sklearn import model_selection

from sklearn.metrics import accuracy_score # Accuracy
classification score
from sklearn.metrics import classification_report # Build a text
report showing the main classification metrics
from sklearn.metrics import confusion_matrix # compute
confusion matrix to evaluate the accuracy of a classification

from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis as LDA # Linear Discriminant Analysis
from sklearn.linear_model import LogisticRegression # Logistic
Regression (aka logit, MaxEnt) classifier
from sklearn.naive_bayes import GaussianNB # Gaussian Naive
Bayes (GaussianNB)
from sklearn.neighbors import KNeighborsClassifier # classifier
implementing the k-nearest neighbors vote
from sklearn.tree import DecisionTreeClassifier # A decision
tree classifier
from sklearn.svc import SVC # C-Support Vector Classification

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/
master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-width',
'class']
dataset = pandas.read_csv(url, names=names)
```

PTO

```
array = dataset.values
X = array[:, 0:4]
Y = array[:, 4]
validation_size = 0.20
seed = 7

X_train, X_validation, Y_train, Y_validation =
        model_selection.train_test_split(X, Y,
                                         test_size = validation_size,
                                         random_state = seed)

models = results = names = []
scoring = 'accuracy'

models.append(('LR', LogisticRegression(solver = 'liblinear',
multi_class = 'ovr')))
models.append(('KNN', KNeighborsClassifier()))
models.append(('CART', DecisionTreeClassifier()))
models.append(('NB', GaussianNB()))
models.append(('SVM', SVC(kernel = 'linear', random_state = 0)))
models.append(('LDA', LDA(n_components = 1)))

for name, model in models:
    kfold = model_selection.KFold(n_splits = 10, random_state = seed,
shuffle = True)
    cv_results = model_selection.cross_val_score(model, X_train,
Y_train, cv= kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(),
cv_results.std())
    print(msg)
```

(c) Analysis of Machine Learning Models using Box and whisker diagrams:

① Import matplotlib.pyplot as plt, a collection of command style functions that make matplotlib like MATLAB.

② Import pandas, library for working with data sets. Import scatter_matrix from pandas plotting to draw a matrix of scatter plots.

③ From sklearn import model_selection, split arrays or matrices into random train and test subsets.

④ Import metrics for classification technique and other model building libraries.

⑤ Load dataset and read the comma-segerated values (csv) file into DataFrame.

⑥ Create training and validation datasets and return a NumPy representation of the DataFrame. Assume data (independent) and target (dependent) variable.

⑦ Control the shuffling applied to the data frame before applying the split and set the scoring criteria.

⑧ Build all the models first, and then evaluate each model.

⑨ Provide train/test indices to split data into train/test sets and evaluate a score by cross-validation.

⑩ Plot Model Results-

(i) Create a new figure, or activate an existing figure.
(ii) Add a centered suptitle to the figure.
(iii) Add an axes to the current figure or retrieve an existing axes
(iv) Make a box and whisker plot
(v) Set the xaxis' labels with list of string labels.
(vi) Display all open figures.

⑪ Machine Learning Model Analysis -

(i) Call the classifier with required parameters.
(ii) Fit model according to the given training data and parameters.
(iii) Predict class labels for samples in X.
(iv) Print accuracy score and classification report.

→ Python Code -

```
import matplotlib.pyplot as plt
import pandas
from pandas.plotting import scatter_matrix
from sklearn import model_selection

from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix

from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
as LDA
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/
master/iris.csv"
names = ['sepal-length', 'sepal-width', 'petal-length', 'petal-
width', 'class']
dataset = pandas.read_csv(url, names=names)

array = dataset.values
X = array[:,0:4]
Y = array[:,4]
validation_size = 0.20
seed = 7

X_train, X_validation, Y_train, Y_validation =
    model_selection.train_test_split(X, Y,
                                     test_size = validation_size,
                                     random_state = seed)
```

```
models= results = names = []
scoring = 'accuracy'

models.append (('LR', LogisticRegression(solver = 'liblinear',
multi_class = 'ovr')))
models.append (('KNN', KNeighborsClassifier()))
models.append (('CART', DecisionTreeClassifier()))
models.append (('NB', GaussianNB()))
models.append (('SVM', SVC(kernel = 'linear', random_state = 0)))
models.append (('LDA', LDA(n_components = 1)))

for name, model in models:
    kfold = model_selection.KFold(n_splits = 10, random_state = seed,
shuffle = True)
    cv_results = model_selection.cross_val_score(model, X_train,
Y_train, cv = kfold, scoring = scoring)
    results.append(cv_results)
    names.append(name)
    msg = "%s: %f (%f)" % (name, cv_results.mean(),
cv_results.std())
    print(msg)

figure = plt.figure()
figure.suptitle('Algorithm Comparison')
algplot = figure.add_subplot(1, 1, 1)
plt.boxplot(results)
algplot.set_xticklabels(names)
plt.show()

print("\n Linear Discriminant Analysis Classifier: \n")
LDA = LDA(n_components = 1)
LDA.fit(X_train, Y_train)
predictions = LDA.predict(X_validation)
print(accuracy_score(Y_validation, predictions))
print(classification_report(Y_validation, predictions))
```

```
print ("\n Logistic Regression classifier : \n")
LR = LogisticRegression (solver = 'liblinear', multi_class = 'ovr')
LR.fit (X_train, Y_train)
predictions = LR.predict (X_validation)
print (accuracy_score (Y_validation, predictions))
print (classification_report (Y_validation, predictions))

print ("\n K-Neighbors classifier : \n")
knn = KNeighborsClassifier ()
knn.fit (X_train, Y_train)
predictions = knn.predict (X_validation)
print (accuracy_score (Y_validation, predictions))
print (classification_report (Y_validation, predictions))

print ("\n Decision Tree classifier : \n")
DT = DecisionTreeClassifier ()
DT.fit (X_train, Y_train)
predictions = DT.predict (X_validation)
print (accuracy_score (Y_validation, predictions))
print (classification_report (Y_validation, predictions))

print ("\n Naive Bayes classifier : \n")
NB = GaussianNB ()
NB.fit (X_train, Y_train)
predictions = NB.predict (X_validation)
print (accuracy_score (Y_validation, predictions))
print (classification_report (Y_validation, predictions))

print ("\n Support Vector Machine (svm) classifier : \n")
SVM = SVC (kernel = 'linear', random_state = 0)
svm.fit (X_train, Y_train)
predictions = svm.predict (X_validation)
print (accuracy_score (Y_validation, predictions))
print (classification_report (Y_validation, predictions))
```

```
LR: 0.958333(0.055902)
KNN: 0.983333(0.033333)
CART: 0.950000(0.076376)
GNB: 0.966667(0.040825)
LDA: 0.975000(0.038188)
SVM: 0.983333(0.033333)
0.9
[[ 7  0  0]
 [ 0 11  1]
 [ 0  2  9]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	7
Iris-versicolor	0.85	0.92	0.88	12
Iris-virginica	0.90	0.82	0.86	11
accuracy			0.90	30
macro avg	0.92	0.91	0.91	30
weighted avg	0.90	0.90	0.90	30

Figure 5 - classifier Results

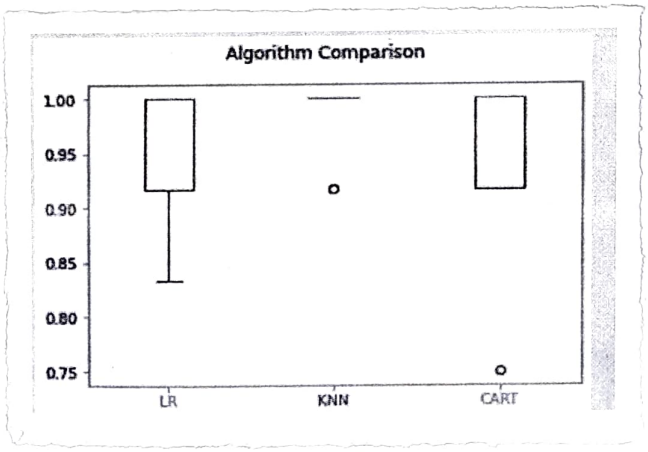

Figure 6 - comparison of LR, KNN and CART results

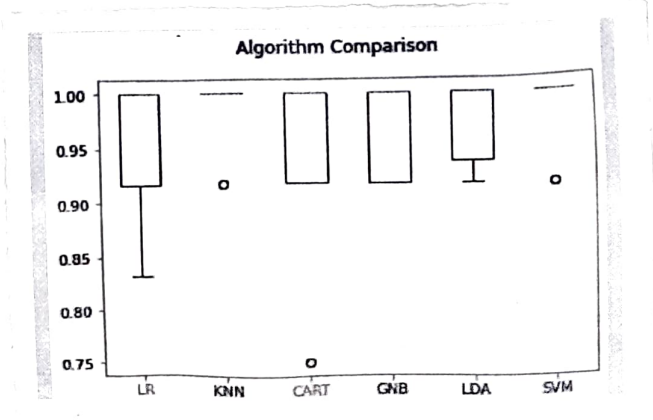

Figure 7 - Comparison of all model results.

RESULT :

Thus, implemented various machine learning techniques using Raspberry Pi. All the simulation results were verified successfully.