

VIT BHOPAL UNIVERSITY

School of Computing Science and Engineering

Bhopal-Indore Highway, Kothrikalan, Sehore

Madhya Pradesh - 466114

CSA4008 - APPLIED MACHINE LEARNING

REG.NO : 23BAI10165

NMAE : Suyash Soni

BRANCH : CSE – AI & ML

SEMESTER: Fall Semester 2025-26

INDEX

|  |  |  |  |
| --- | --- | --- | --- |
| Ex.NO | DATE | EXPERIMENT NAME | PAGE NO. |
|  |  | **Implement Decision Tree learning** |  |
|  |  | **Implement Logistic Regression** |  |
|  |  | **Implement classification using Multilayer perceptron** |  |
|  |  | **Implement classification using Multilayer perceptron** |  |
|  |  | **Implement Adaboost Algorithm** |  |
|  |  | **Implement Bagging using Random Forests** |  |
|  |  | **Implement K-means Clustering to Find Natural Patterns in Data** |  |
|  |  | **Implement Principle Component Analysis for Dimensionality Reduction** |  |
|  |  | **Evaluating ML algorithm with balanced and unbalanced datasets** |  |
|  |  | **Evaluating ML algorithm with balanced and unbalanced datasets** |  |

|  |  |
| --- | --- |
| **EXP.NO: 03** | **Implement classification using Multilayer perceptron** |
| **DATE:** |

**AIM : Implement classification using Multilayer perceptron**

**PROGRAM**

import random

import seaborn

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import matplotlib.cm as cm

seaborn.set(style='whitegrid'); seaborn.set\_context('talk')

%matplotlib inline

%config InlineBackend.figure\_format = 'retina'

from sklearn.datasets import load\_iris

iris\_data = load\_iris()

print(iris\_data['DESCR'])

n\_samples, n\_features = iris\_data.data.shape

plt.subplot(1, 2, 1)

scatter\_plot = plt.scatter(iris\_data.data[:,0], iris\_data.data[:,1], alpha=0.5,

c=iris\_data.target)

plt.colorbar(ticks=([0, 1, 2]))

plt.title('Sepal Sample')

plt.subplot(1, 2, 2)

scatter\_plot\_2 = plt.scatter(iris\_data.data[:,2], iris\_data.data[:,3], alpha=0.5,

c=iris\_data.target)

plt.colorbar(ticks=([0, 1, 2]))

plt.title('Petal Sample')

import pandas

from pandas.plotting import scatter\_matrix

dataset = pandas.read\_csv('../input/iris/Iris.csv')

scatter\_matrix(dataset, alpha=0.5, figsize=(20, 20))

plt.show()

dataset.hist(alpha=0.5, figsize=(20, 20), color='red')

plt.show()

dataset.plot(subplots=True, figsize=(10, 10), sharex=False, sharey=False)

plt.show()

random.seed(123)

def separate\_data():

A = iris\_dataset[0:40]

tA = iris\_dataset[40:50]

B = iris\_dataset[50:90]

tB = iris\_dataset[90:100]

C = iris\_dataset[100:140]

tC = iris\_dataset[140:150]

train = np.concatenate((A,B,C))

test = np.concatenate((tA,tB,tC))

return train,test

train\_porcent = 80 # Porcent Training

test\_porcent = 20 # Porcent Test

iris\_dataset = np.column\_stack((iris\_data.data,iris\_data.target.T)) #Join X and Y

iris\_dataset = list(iris\_dataset)

random.shuffle(iris\_dataset)

Filetrain, Filetest = separate\_data()

train\_X = np.array([i[:4] for i in Filetrain])

train\_y = np.array([i[4] for i in Filetrain])

test\_X = np.array([i[:4] for i in Filetest])

test\_y = np.array([i[4] for i in Filetest])

import matplotlib.pyplot as plt

import matplotlib.cm as cm

plt.subplot(1, 2, 1)

plt.scatter(train\_X[:,0],train\_X[:,1],c=train\_y,cmap=cm.viridis)

plt.xlabel(iris\_data.feature\_names[0])

plt.ylabel(iris\_data.feature\_names[1])

plt.subplot(1, 2, 2)

plt.scatter(train\_X[:,2],train\_X[:,3],c=train\_y,cmap=cm.viridis)

plt.xlabel(iris\_data.feature\_names[2])

plt.ylabel(iris\_data.feature\_names[3])

plt.subplot(1, 2, 1)

plt.scatter(test\_X[:,0],test\_X[:,1],c=test\_y,cmap=cm.viridis)

plt.xlabel(iris\_data.feature\_names[0])

plt.ylabel(iris\_data.feature\_names[1])

plt.subplot(1, 2, 2)

plt.scatter(test\_X[:,2],test\_X[:,3],c=test\_y,cmap=cm.viridis)

plt.xlabel(iris\_data.feature\_names[2])

plt.ylabel(iris\_data.feature\_names[3])

x = 0

ativation = {(lambda x: 1/(1 + np.exp(-x)))}

deriv = {(lambda x: x\*(1-x))}

activation\_tang = {(lambda x: np.tanh(x))}

deriv\_tang = {(lambda x: 1-x\*\*2)}

activation\_ReLU = {(lambda x: x\*(x > 0))}

deriv\_ReLU = {(lambda x: 1 \* (x>0))}

graph\_hits = []

print("Porcents :","%.2f"%(hits),"% hits","and","%.2f"%(faults),"% faults")

print("Total samples of test",n\_samples)

print("\*Iris-Setosa:",n\_set,"samples")

print("\*Iris-Versicolour:",n\_vers,"samples")

print("\*Iris-Virginica:",n\_virg,"samples")

graph\_hits.append(hits)

graph\_hits.append(faults)

labels = 'Hits', 'Faults';

sizes = [96.5, 3.3]

explode = (0, 0.14)

fig1, ax1 = plt.subplots();

ax1.pie(graph\_hits, explode=explode,colors=['green','red'],labels=labels, autopct='%1.1f%%',

shadow=True, startangle=90)

ax1.axis('equal')

plt.show()

acc\_set = (score\_set/n\_set)\*100

acc\_vers = (score\_vers/n\_vers)\*100

acc\_virg = (score\_virg/n\_virg)\*100

print("- Acurracy Iris-Setosa:","%.2f"%acc\_set, "%")

print("- Acurracy Iris-Versicolour:","%.2f"%acc\_vers, "%")

print("- Acurracy Iris-Virginica:","%.2f"%acc\_virg, "%")

names = ["Setosa","Versicolour","Virginica"]

x1 = [2.0,4.0,6.0]

fig, ax = plt.subplots()

r1 = plt.bar(x1[0], acc\_set,color='orange',label='Iris-Setosa')

r2 = plt.bar(x1[1], acc\_vers,color='green',label='Iris-Versicolour')

r3 = plt.bar(x1[2], acc\_virg,color='purple',label='Iris-Virginica')

plt.ylabel('Scores %')

plt.xticks(x1, names);plt.title('Scores by iris flowers - Multilayer Perceptron')

plt.show()

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Generate confusion matrix

cm = confusion\_matrix(test\_y, prev)

labels = ['Setosa', 'Versicolour', 'Virginica']

plt.figure(figsize=(6, 5))

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues', xticklabels=labels, yticklabels=labels)

plt.title('Confusion Matrix')

plt.xlabel('Predicted')

plt.ylabel('True')

plt.show()

import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

# Create DataFrame from test set for correlation

df\_corr = pd.DataFrame(test\_X, columns=iris\_data.feature\_names)

plt.figure(figsize=(8, 6))

sns.heatmap(df\_corr.corr(), annot=True, fmt=".2f", cmap="coolwarm", square=True)

plt.title("Feature Correlation Matrix - Iris Dataset")

plt.tight\_layout()

plt.show()

from sklearn.metrics import confusion\_matrix

import seaborn as sns

import matplotlib.pyplot as plt

# Get predictions again

y\_pred, \_ = Perceptron.predict(test\_X, test\_y)

# Generate confusion matrix

cm = confusion\_matrix(test\_y, y\_pred)

labels = ['Setosa', 'Versicolour', 'Virginica']

# Plot confusion matrix heatmap

plt.figure(figsize=(6,5))

sns.heatmap(cm, annot=True, cmap='Blues', xticklabels=labels, yticklabels=labels, fmt='d')

plt.xlabel('Predicted Labels')

plt.ylabel('True Labels')

plt.title('Confusion Matrix - MLP Iris Classification')

plt.tight\_layout()

plt.show()

import numpy as np

labels = ['Setosa', 'Versicolour', 'Virginica']

scores = [acc\_set, acc\_vers, acc\_virg]

colors = ['orange', 'green', 'purple']

plt.figure(figsize=(7,4))

bars = plt.bar(labels, scores, color=colors)

plt.ylabel('Accuracy %')

plt.ylim(0, 105)

plt.title('MLP Accuracy by Iris Class')

# Add percentage labels on top

for bar in bars:

yval = bar.get\_height()

plt.text(bar.get\_x() + bar.get\_width()/2, yval + 1, f'{yval:.1f}%', ha='center')

plt.tight\_layout()

plt.show()

import matplotlib.pyplot as plt

import numpy as np

# Assuming you already calculated these values

class\_labels = ['Setosa', 'Versicolour', 'Virginica']

correct = [score\_set, score\_vers, score\_virg] # From your model's prediction analysis

total = [n\_set, n\_vers, n\_virg] # Total samples per class

# Calculate incorrect predictions

incorrect = [total[i] - correct[i] for i in range(3)]

accuracy = [(correct[i] / total[i]) \* 100 for i in range(3)]

# Plotting

x = np.arange(len(class\_labels))

bar\_width = 0.6

fig, ax = plt.subplots(figsize=(8, 6))

# Bars

bars\_correct = ax.bar(x, correct, bar\_width, label='Correct', color='green')

bars\_incorrect = ax.bar(x, incorrect, bar\_width, bottom=correct, label='Incorrect', color='red')

# Accuracy annotations

for i in range(len(class\_labels)):

ax.text(x[i], total[i] + 0.5, f"{accuracy[i]:.1f}%", ha='center', fontsize=12, fontweight='bold')

# Axes & Labels

ax.set\_ylabel('Number of Samples')

ax.set\_title('MLP Classification Results per Iris Class')

ax.set\_xticks(x)

ax.set\_xticklabels(class\_labels)

ax.legend()

ax.set\_ylim(0, max(total) + 10)

plt.tight\_layout()

plt.show()

**INPUT**

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2518 entries, 0 to 2517

Data columns (total 13 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Date 2518 non-null datetime64[ns]

1 Close/Last 2518 non-null float64

2 Volume 2518 non-null int64

3 Open 2518 non-null float64

4 High 2518 non-null float64

5 Low 2518 non-null float64

6 Price\_Up 2518 non-null int32

7 Price\_Range 2518 non-null float64

8 Price\_Change 2518 non-null float64

9 Price\_Change\_Pct 2518 non-null float64

10 Volume\_MA\_5 2514 non-null float64

11 Close\_MA\_5 2514 non-null float64

12 Volume\_Above\_Avg 2518 non-null int32

dtypes: datetime64[ns](1), float64(9), int32(2), int64(1)

memory usage: 236.2 KB

------------------------------------------------------------------------------

Dataset shape: (2514, 6)

Target distribution:

Price\_Up

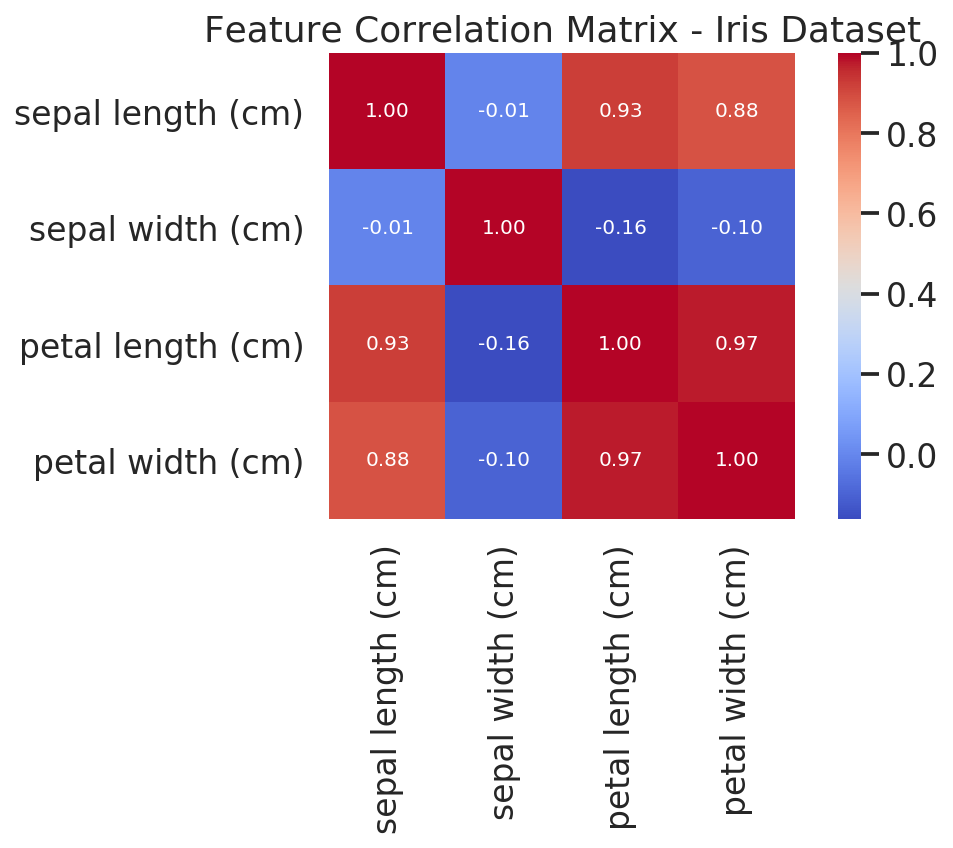
0 1332

1 1182

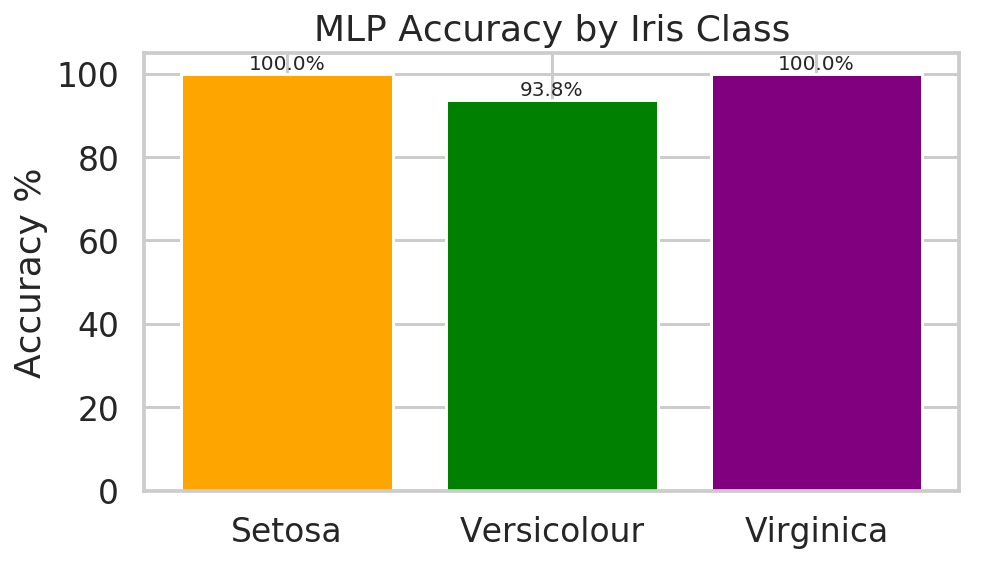
Name: count, dtype: int64

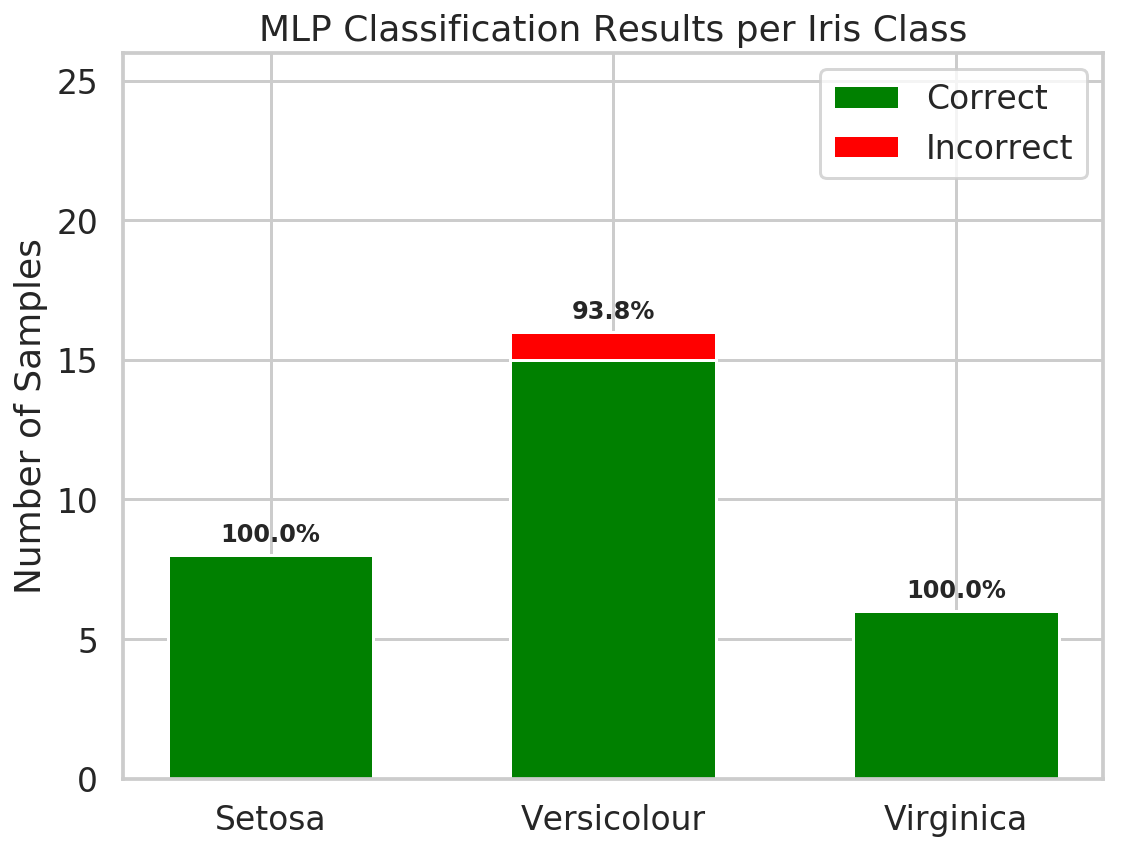
**OUTPUT**

----------------------------------------------------------------------------------









**RESULT**

Porcents : 96.67 % hits and 3.33 % faults

Total samples of test 150

\*Iris-Setosa: 8 samples

\*Iris-Versicolour: 16 samples

\*Iris-Virginica: 6 samples

