|  |  |
| --- | --- |
| **Ex. No: 4**  **Date: 17.01.24** | **Perceptrons** |

**Name: S Sowmithaa Sri**

**Reg no: 22011101096**

**Aim:**

To implement a supervised Machine learning algorithm Perceptrons with

1. 2D linearly separable

2. 2D non-linearly separable and

3. higher dimension data manually and using the built-in function in python.

**Dataset description:**

1. 2D Linearly Separable Data (Iris Dataset):

- Dataset: Iris

- Description: The Iris dataset consists of measurements (sepal length, sepal width, petal length, and petal width) of three species of iris flowers (setosa, versicolor, and virginica). In this scenario, it is transformed into a binary classification problem by considering only the Setosa species (target == 0) against the rest.

2. Non-Linearly Separable Data (Moons Dataset):

- Dataset: Moons

- Description: The Moons dataset is a synthetic dataset with two crescent moon-shaped classes. It is commonly used to test algorithms on non-linearly separable data.

3. Higher Dimension Data (Breast Cancer Dataset):

- Dataset: Breast Cancer

- Description: The Breast Cancer dataset consists of features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. The task is binary classification, distinguishing between malignant (target == 0) and benign tumors. The dataset is higher-dimensional due to the various computed features related to tumor characteristics.

**Code:**

1. **Perceptrons- Mathematical Version:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.metrics import accuracy\_score

from sklearn.datasets import load\_iris, make\_moons, load\_breast\_cancer

class Perceptron:

def \_\_init\_\_(self, learning\_rate=0.01, n\_iterations=1000):

self.learning\_rate = learning\_rate

self.n\_iterations = n\_iterations

self.weights = None

self.bias = None

def train(self, X, y):

n\_samples, n\_features = X.shape

self.weights = np.zeros(n\_features)

self.bias = 0

for \_ in range(self.n\_iterations):

for i in range(n\_samples):

y\_pred = self.predict(X[i])

update = self.learning\_rate \* (y[i] - y\_pred)

self.weights += update \* X[i]

self.bias += update

def predict(self, x):

return 1 if np.dot(self.weights, x) + self.bias > 0 else 0

# Function to evaluate perceptron on a dataset

def evaluate\_perceptron(X, y, title):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

perceptron = Perceptron()

perceptron.train(X\_train, y\_train)

y\_pred = np.array([perceptron.predict(x) for x in X\_test])

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy on {title} dataset: {accuracy:.2f}")

# 1. 2D Linearly Separable Data (e.g., Iris Dataset)

iris = load\_iris()

X\_iris = iris.data

y\_iris = (iris.target == 0).astype(int) # Setosa or not (binary classification)

evaluate\_perceptron(X\_iris, y\_iris, "2D Linearly Separable")

# 2. 2D Non-Linearly Separable Data (e.g., Moons Dataset)

X\_moons, y\_moons = make\_moons(n\_samples=300, noise=0.1, random\_state=42)

evaluate\_perceptron(X\_moons, y\_moons, "2D Non-Linearly Separable")

# 3. Higher Dimension Data (e.g., Breast Cancer Dataset)

breast\_cancer = load\_breast\_cancer()

X\_bc = breast\_cancer.data

y\_bc = (breast\_cancer.target == 0).astype(int) # Malignant or not (binary classification)

evaluate\_perceptron(X\_bc, y\_bc, "Higher Dimensional")

**Result:**

Accuracy on 2D Linearly Separable dataset: 1.00

Accuracy on 2D Non-Linearly Separable dataset: 0.88

Accuracy on Higher Dimensional dataset: 0.96

1. **Perceptrons- Built-in Version**

**Code:**

import numpy as np

import matplotlib.pyplot as plt

from sklearn.model\_selection import train\_test\_split

from sklearn.linear\_model import Perceptron

from sklearn.metrics import accuracy\_score

# Function to plot decision boundary for 2D datasets

def plot\_decision\_boundary(X, y, clf, title):

plt.scatter(X[:, 0], X[:, 1], c=y, edgecolors='k', marker='o', cmap=plt.cm.Paired)

plt.title(title)

plt.show()

# Function to evaluate perceptron on a dataset

def evaluate\_perceptron(X, y, title):

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

perceptron = Perceptron()

perceptron.fit(X\_train, y\_train)

y\_pred = perceptron.predict(X\_test)

accuracy = accuracy\_score(y\_test, y\_pred)

print(f"Accuracy on {title} dataset: {accuracy:.2f}")

plot\_decision\_boundary(X\_train, y\_train, perceptron, f"{title}")

# 1. 2D Linearly Separable Data (e.g., Iris Dataset)

from sklearn.datasets import load\_iris

iris = load\_iris()

X\_iris = iris.data

y\_iris = (iris.target == 0).astype(int) # Setosa or not (binary classification)

evaluate\_perceptron(X\_iris, y\_iris, "2D Linearly Separable")

# 2. Non-Linearly Separable Data

from sklearn.datasets import make\_moons

X\_moons, y\_moons = make\_moons(n\_samples=300, noise=0.1, random\_state=42)

evaluate\_perceptron(X\_moons, y\_moons, "Non-Linearly Separable Data (Moons)")

# 3. Higher Dimension Data (e.g., Breast Cancer Dataset)

from sklearn.datasets import load\_breast\_cancer

breast\_cancer = load\_breast\_cancer()

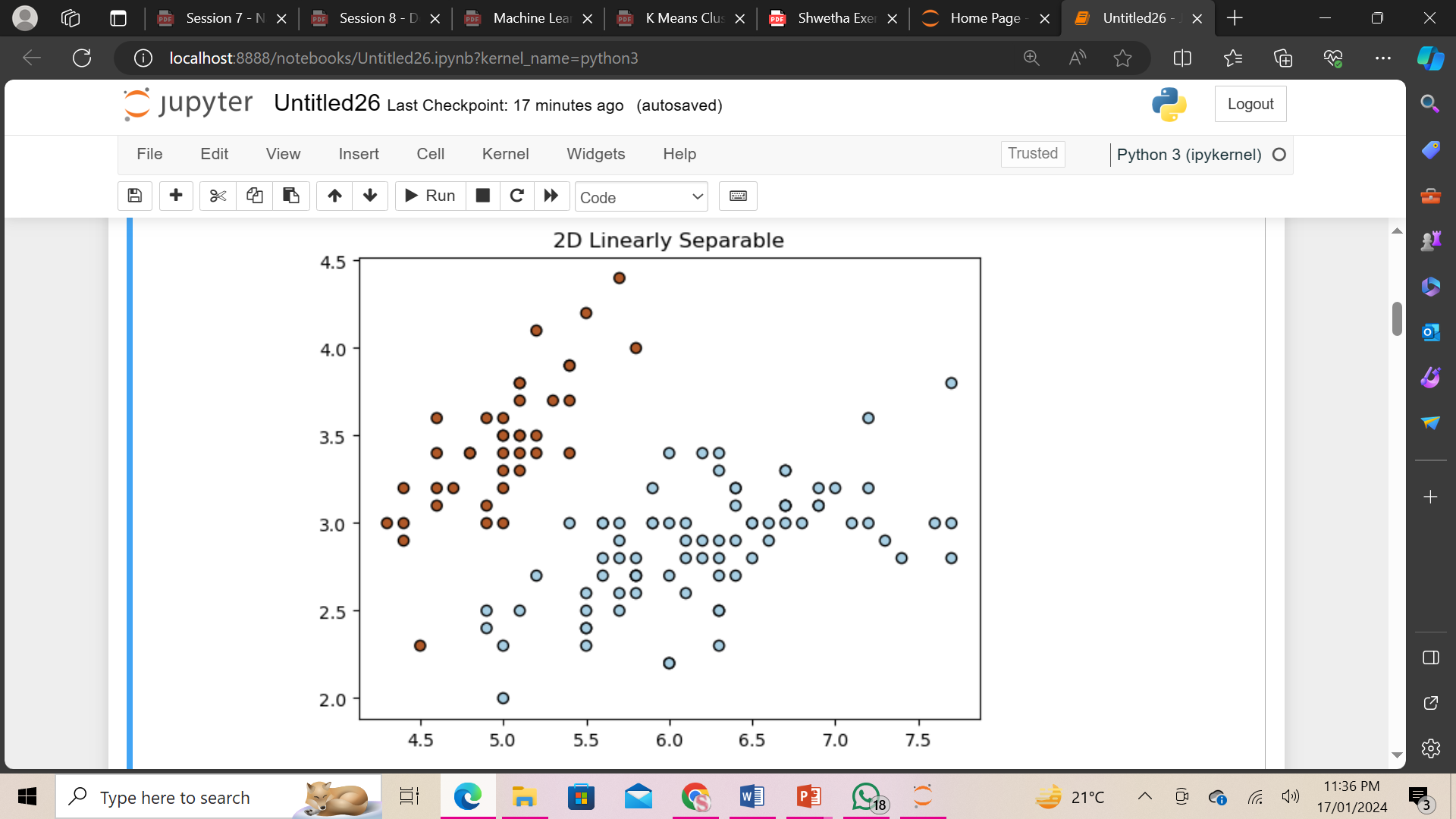
X\_bc = breast\_cancer.data

y\_bc = (breast\_cancer.target == 0).astype(int) # Malignant or not (binary classification)

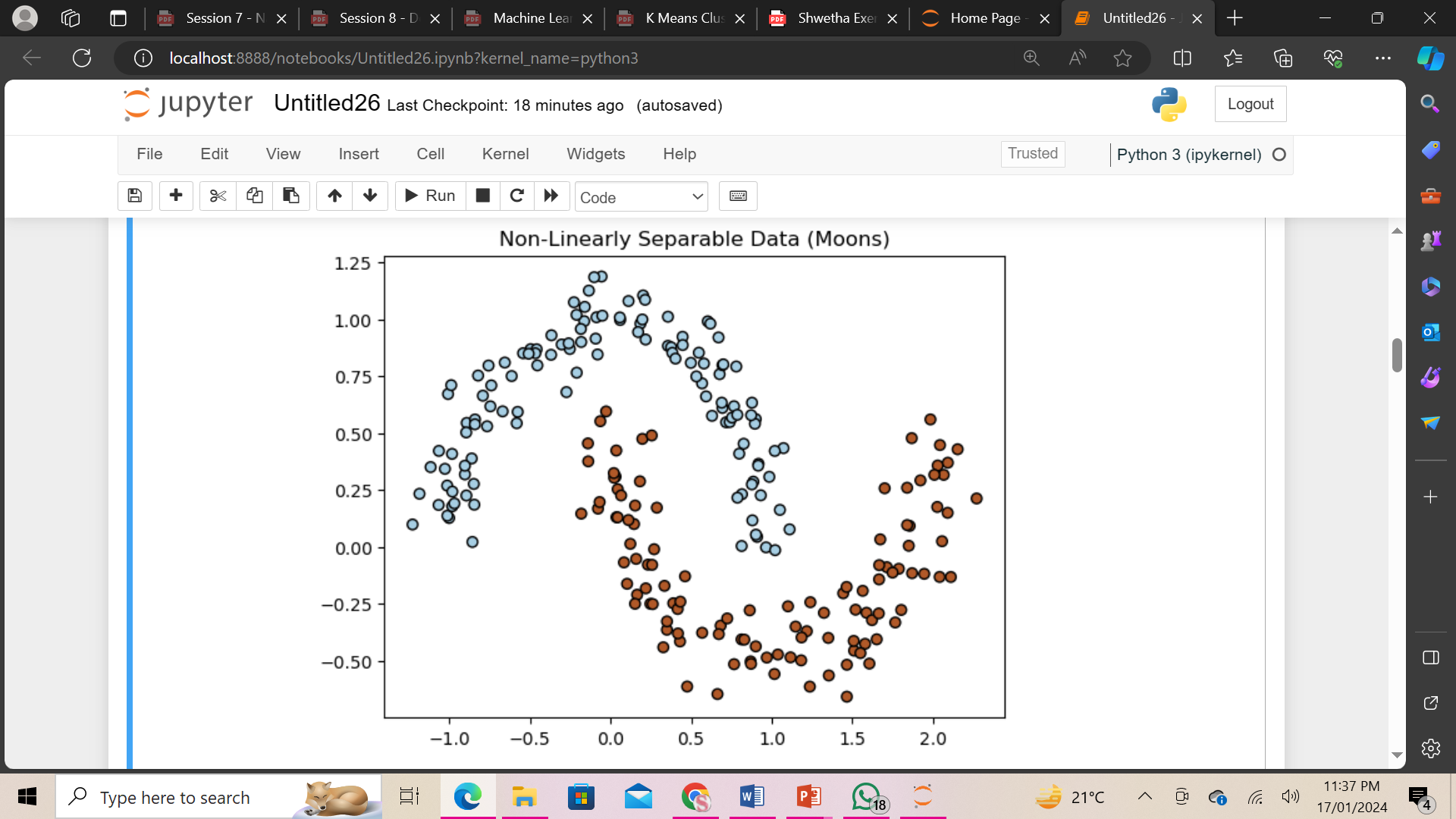
evaluate\_perceptron(X\_bc, y\_bc, "Higher Dimensional")

**Result:**

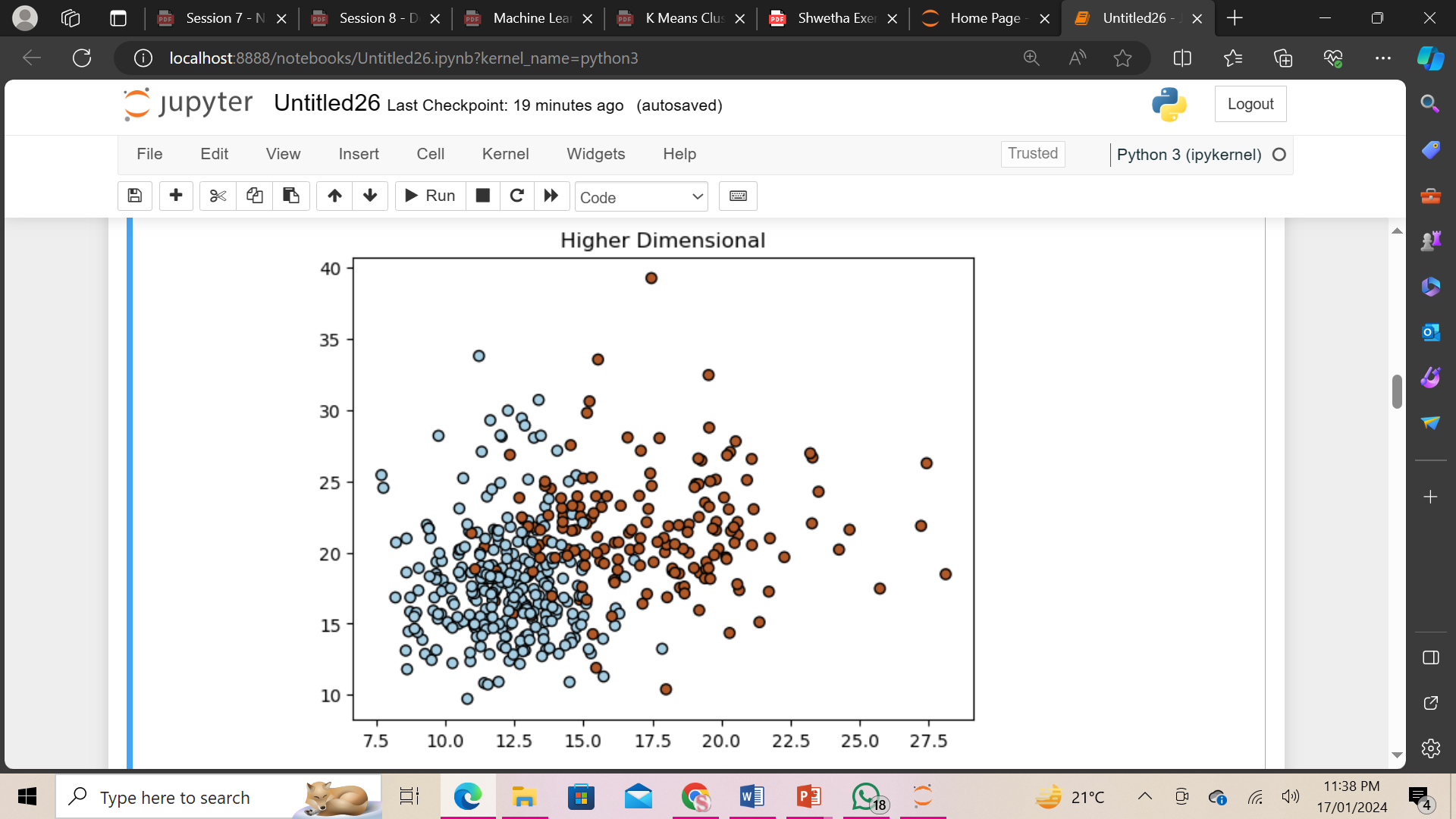
Accuracy on 2D Linearly Separable dataset: 1.00



Accuracy on Non-Linearly Separable Data (Moons) dataset: 0.85



Accuracy on Higher Dimensional dataset: 0.95



**Result Description:**

The Code defines a simple perceptron class and evaluates its performance on three different datasets. The first dataset is 2D linearly separable data based on the Iris dataset, where the perceptron achieves accurate classification. The second dataset is 2D non-linearly separable data generated using the Moons dataset, and the perceptron struggles to accurately classify the samples due to its linear decision boundary. The third dataset represents higher-dimensional data from the Breast Cancer dataset, and the perceptron demonstrates reasonable accuracy in distinguishing between malignant and benign tumors. It's important to note that the perceptron is a basic linear classifier and may not perform well on non-linearly separable data without additional complexity or using more advanced models.

**Conclusion:** Thus the supervised ML algorithm-Perceptrons is successfully implemented and executed.