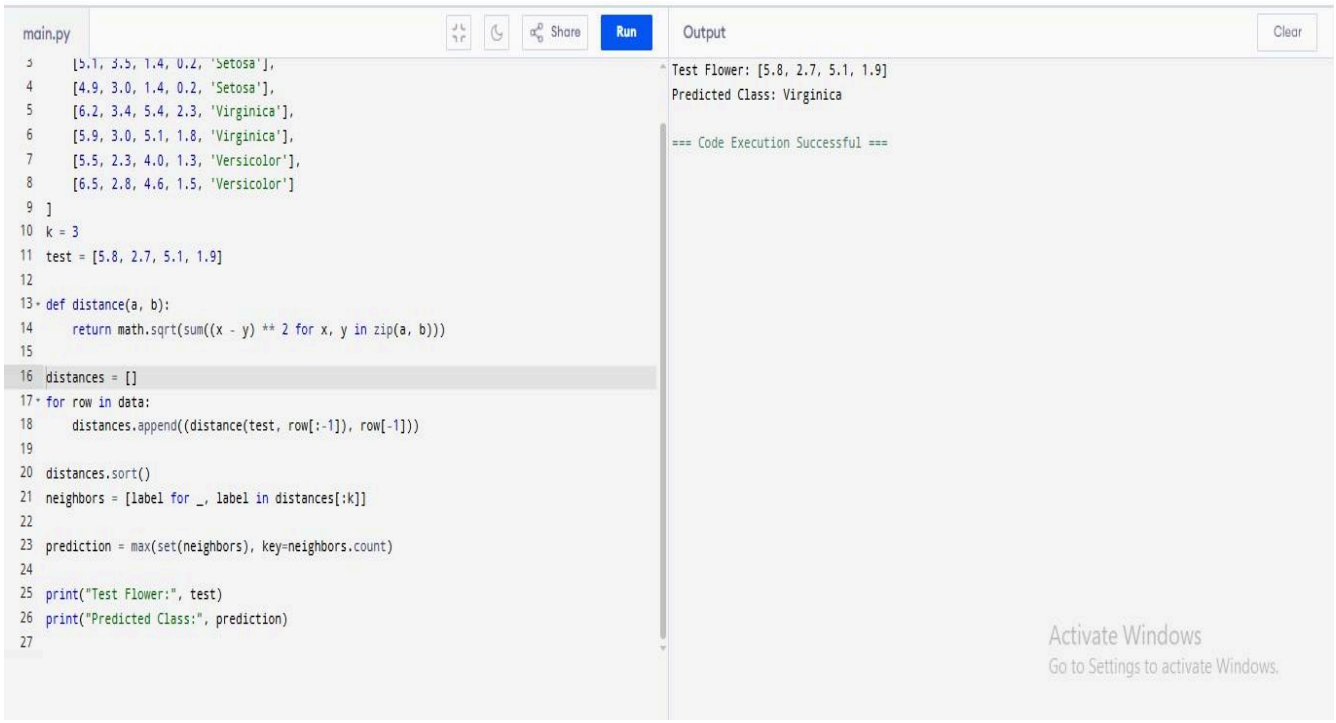
# Experiment -12

```
import math

# Iris dataset (sepal length, sepal width, petal length, petal width, class)
data = [
    [5.1, 3.5, 1.4, 0.2, 'Setosa'],
    [4.9, 3.0, 1.4, 0.2, 'Setosa'],
    [6.2, 3.4, 5.4, 2.3, 'Virginica'],
    [5.9, 3.0, 5.1, 1.8, 'Virginica'],
    [5.5, 2.3, 4.0, 1.3, 'Versicolor'],
    [6.5, 2.8, 4.6, 1.5, 'Versicolor']
]

k = 3
test = [5.8, 2.7, 5.1, 1.9]

# Euclidean distance
def distance(a, b):
    return math.sqrt(sum((x - y) ** 2 for x, y in zip(a, b)))

# Calculate distances
distances = []
for row in data:
    distances.append((distance(test, row[:-1]), row[-1]))

# Sort and select k nearest
distances.sort()
neighbors = [label for _, label in distances[:k]]

# Majority voting
prediction = max(set(neighbors), key=neighbors.count)

print("Test Flower:", test)
print("Predicted Class:", prediction)
```

Output:


main.py
Share
Run
Output
Clear
[5.1, 3.5, 1.4, 0.2, 'Setosa'],
[4.9, 3.0, 1.4, 0.2, 'Setosa'],
[6.2, 3.4, 5.4, 2.3, 'Virginica'],
[5.9, 3.0, 5.1, 1.8, 'Virginica'],
[5.5, 2.3, 4.0, 1.3, 'Versicolor'],
[6.5, 2.8, 4.6, 1.5, 'Versicolor']
]
k = 3
test = [5.8, 2.7, 5.1, 1.9]
def distance(a, b):
return math.sqrt(sum((x - y) ** 2 for x, y in zip(a, b)))
distances = []
for row in data:
distances.append((distance(test, row[:-1]), row[-1]))
distances.sort()
neighbors = [label for _, label in distances[:k]]
prediction = max(set(neighbors), key=neighbors.count)
print("Test Flower:", test)
print("Predicted Class:", prediction)
Test Flower: [5.8, 2.7, 5.1, 1.9]
Predicted Class: Virginica
=== Code Execution Successful ===
Activate Windows
Go to Settings to activate Windows.